Title:	*MODAClouds Integrated Solution - Final Version*
Authors:	*Daniel Pop, Gabriel Iuhasz (IeAT), Marcos Almeida, Anthonin Abhervé (Softeam), Michele Ciavotta (Polimi,) Darren Whigham(Flex), Jacek Dominiak (CA), Nicolas Ferry (SINTEF), Rasha Osman, Wang Weikun, Pooyan Jamshidi (IMP)*
Editor:	*Craig Sheridan (Flexi)*
Reviewers:	*Danilo ARDAGNA (Polimi)*
	Nicolas FERRY (SINTEF)
Identifier:	*Deliverable D3.5.3*
Nature:	*Prototype*
Version:	*1*
Date:	*07/10/2015*
Status:	*Final*
Diss. level:	*Public*

Executive Summary

This document (deliverable D3.5.3) describes the technologies and environment that represents the final delivery of the MODAClouds integrated solution (deliverable D3.5.3). It documents the implementation of the architecture of the MODAClouds solution detailed in D3.3.2, which defined the necessary interfaces to integrate the various components.

Copyright © 2015 by the MODAClouds consortium – All rights reserved.

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 318484 (MODAClouds).

Members of the MODAClouds consortium:

Politecnico di Milano	Italy
StiftelsenSintef	Norway
Institutul E-Austria Timisoara	Romania
Imperial College of Science, Technology and Medicine	United Kingdom
SOFTEAM	France
Siemens Program and System Engineering	Romania
BOC Information Systems GMBH	Austria
Flexiant Limited	United Kingdom
ATOS Spain S.A.	Spain
CA Technologies Development Spain S.A.	Spain

Published MODAClouds documents

These documents are all available from the project website located at http://www.modaclouds.eu/

Contents

1 INTRODUCTION 5

1.1 CONTEXT AND OBJECTIVES 5
1.2 STRUCTURE OF THE DOCUMENT 5

2 OVERVIEW OF THE INTEGRATION PROTOTYPE 7

2.1 VENUES 4CLOUDS 7
2.2 CREATOR 4CLOUDS 8
Functional Modelling Tool 11
CPIM library 11
SpaceDev 4Clouds 11
LINE 11
CloudML 11
Resource Repository 12
Feedback Loop 12
2.3 ENERGIZER4CLOUDS 12
ADDapters 4Clouds 12
Data Migration 13
Load Balancer Controller 13
Object Store 13
Artifact Repository 13
mOS Image 14
mOS Package Builder 14
Tower 4Clouds 14
RDF History DB 16
SpaceOps 4Clouds 16

3 M36 INTEGRATED PLATFORM TESTBEDS 18

3.1 FLEXIANT FCO 18
3.2 IEAT TESTBED 20

4 MODACLOUDS RUNTIME PLATFORM BUILD INSTRUCTIONS 21

4.1 BUILDING RPMS 21
4.2 MANUAL DEPLOYMENT 22
Installation 22
Configuration 23
Running the components 23
Updating from v1.0 to v2.0 23
4.3 AUTOMATIC DEPLOYMENT IN MOSAIC 24

5 CONCLUSION 24

6 REFERENCES 24

A MODACLOUDS INSTALLATION AND USAGE 25

A.1 CREATOR 4CLOUDS INSTALLATION AND USAGE GUIDE 25
Introduction 25
Key Functions of Creator 4Clouds 25
Creator 4Clouds Prerequisites 26
Installing Creator 4Clouds 26
Adding modules from the Modelio modules catalogue 26
Creating a new project 26
Adding a module to a project 27
Adding the Resource Model to a Project 28
Model an application at CCIM level 28
Model an application at CCPM level 29
Model QoS constraints 31
Generate monitoring rules 32
Deploy application 33

Deploy rules *33*
A.2 SPACE 4CLOUDS INSTALLATION AND USAGE GUIDE 34

1 Introduction

1.1 Context and objectives

This document provides a follow up to the MODAClouds platform integration plan deliverables D3.5.1 and D3.5.2. The deliverable is an output of work conducted during both Task 3.3 and 3.4 within WP3.

This is also a companion document to D3.4.2 which reports on the integration of various MODAClouds components and is also due for submission at M36. This third and last integration phase at M36 concludes the 3 planned phases with the first deliverable released at M18 and the second at M30.

This final phase sees MODAClouds software and tools running and hosted on the Flexiant and IEAT infrastructures and progresses and finalises the work delivered at M18 and M30. This final integration phase provides the third and last MODAClouds [1] integrated platform with new improvements and developments relative to the first proof of concept version and initial integrated versions.

The deliverable will show how the MODAClouds prototypes have progressed to an integrated solution as described in Task 3.4 and how the consortium has ably delivered a working solution that fulfills the original objectives and MODAClouds goals. The various individual components have achieved tighter integration to each other where apt and the monitoring and runtime platform finally integrated to the cloud infrastructure of Flexiant and IEAT, converting mere testbeds and stand-alone components, to service offerings with a transparent back-end. This was achieved under the coordination of Task 3.3, with the IEAT and Flexiant infrastructure also used to develop the MODAClouds Supporting Services [5].

The final phase has delivered an innovative MODAClouds framework as envisaged in the DoW and progressed the first iteration of D3.5.1 [6] from proof of concept through to a working and usable cloud service offering.

The deliverable continues the new naming scheme of MODAClouds components, adopted and repackaged to be more marketable together with a distinct branding. The repackaged service offerings that make up the integrated solution can be summarised as:

- **Venues 4Clouds** consisting of Decision Support System (DSS), DSS User Interface, Data collectors, Risk Analysis Engine
- **Creator 4Clouds** consisting of Functional Modelling Tool, **Space^{Dev} 4Clouds**, LINE, CloudML, , Resource Repository.
- **Energizer 4Clouds** consisting of the follow sub-packages
    - **Tower 4Clouds** consisting of Monitoring Manager, DDA, Data Collector Factory, QoS Models, Metrics Observer, Metrics Explorer, Knowledge Base, Data Collectors, Mathlab/Weka SDA.
    - **Space^{Ops} 4Clouds** consisting of Self-Adaptation Reasoner, Self-Adaptation Stress Tester, Load-Balancer Reasoner, Models@Runtime engine.
    - **ADDapters 4Clouds** consisting of Data Migration and Synchronization, Load-Balancer Controller, Object Store, Artefact Repository, Batch Engine, mOS Image Builder, mOS Package Builder.

1.2 Structure of the document

This document is structured as follows:

- Section 1 contextualises the integration status and the relationship of the deliverable to earlier MODAClouds work.
- Section 2 overviews of the final integration prototype broken down to exploitable assets and their sub-components.
- Section 3 describes the current status of the two MODAClouds integrated environments.
- Section 4 explains how to install RPMs and manually deploy and configure MODAClouds tools
- Section 5 concludes the deliverable.

2 Overview of the Integration prototype

The first version of the MODAClouds prototype represented the implementation of the plan described in D3.3.2 [2] and was a working prototype bringing together the effort from various WPs using the Star Integration plan described in the proof of concept deliverable [6]. The initial integration plan was then documented in D3.5.2. This final integration deliverable is an extended version of the MODAClouds design-time and run-time platform, which is hosted, mainly, on the Flexiant infrastructure and integrates the components described in the D3.2.2 architecture document [3] to the execution platform.

There are three major assets, which have been integrated to the execution platform to constitute the overall MODAClouds framework, Creator 4Clouds, Venues4Clouds and Energizer4Clouds, the latter composed of Tower4Clouds, ADDapter 4Clouds and Space^{Ops} 4Clouds.

Creator 4Clouds asset is the Design-Time Platform provided by the MODAClouds Toolbox. It is a multi-cloud modelling & deployment integrated development environment, which allows users to design cloud applications, identify components and key functions of applications, and define the desired QoS of the application.

Venues 4Clouds is a decision support system of the cloud application being modelled in Creator 4Clouds. This component allows the users to express their requirements, processes these requirements via risk-based analysis and presents a list of sets of cloud services that are most suitable to meet those requirements.

Once the application has been modelled, it can be deployed and managed in the clouds using Energizer 4Clouds. ADDapters4Clouds asset is responsible for connecting the MODAClouds Platform to several underlying cloud service providers at IaaS and PaaS levels. Tower4Clouds asset is responsible for providing a monitoring platform that keeps track of the performance of the application and evaluates the QoS constraints defined at design-time in Creator 4Clouds. Space^{Ops} 4Clouds asset manages the lifecycle of the application, shows the status of the multi-cloud application and its execution environment to the operators, and is able to perform certain reconfiguration actions depending on the metrics provided by Tower 4Clouds.

2.1 Venues 4Clouds

Venues 4Clouds is the decision support system of MODAClouds solution. This component allows the users to express their requirements, processes these requirements via risk-based analysis and presents a list of sets of cloud services that are most suitable to meet those requirements.
It relies on generic data gathering mechanisms to collect the information about the characteristics of the cloud marketplace and systematic processing of users requirements into concrete and comparable characteristics of the cloud services.

Exploitable Asset	**Module**	**Component**	**Integrated or External service**
Venues 4Clouds	***Decision Support System (DSS)***	DSS UI	Integrated
		Risk Analysis	External
		Data collectors- Dss-data-import	Integrated
		Data collectors- Dss-data-save	Integrated

Table 1: Venues4Clouds Components

It consists of User Interface in order to gather different sets of requirements, data collectors and a risk analysis engine. User Interface guides the user though the process of gathering different sets of requirements, technical as well as business oriented, assesses them and helps to define a matrix of potential risks with associated likelihood and consequence. Furthermore, it proposes potential treatments to mitigate the risks. In addition, User Interface provides different sets of visualizations to help the user comprehend underlying complexities in the cloud services arena. Data collectors are sophisticated mechanisms to acquire, process and feed the data from multiple sources like, API's, websites etc. into the DSS. Risks analysis engine takes input in the form of business and

technical requirements from the user and conducts the risk analysis based approach to formulate set of cloud services characteristics that meet the user requirements ensuring risk mitigation.
The tree components of the DSS function together to provide an efficient service match making that recommends the users the set of services for required cloud types.

DSS User Interface

Code: https://github.com/CA-Labs/DSS
Documentation: D.2.3.2
Deliverable: D2.3.2
License: Apache Version 2.0
Web-based application, which though the process of 6 steps guidance allows the user to express the requirements, assess the risks and select the treatments. Each step takes into account different dimension of cloud provider selection and helps the user to identify the exact characteristics of the provider.

Data collectors (Dss-data-import, Dss-data-save)

Code: https://github.com/CA-Labs/dss-data-import
Documentation: README file
Deliverable: D2.3.2
License: Apache Version 2.0
Acquires the data from multiple heterogeneous sources and transforms to common format that can be fed directly into DSS data store. The module is designed as a standalone CLI-based application and can be implemented across different types of uses, not necessary connected to the Cloud Service Provider scenarios. It can consume any type of web or local structured data source.

Risk Analysis Engine

Code: https://github.com/CA-Labs/DSS
Documentation: D2.3.2
Deliverable: D.2.1.2
License: Apache Version 2.0
As an internal part of the DSS User Interface it allows the user to specify the risks and the values of the risks that can be faced by the deployment in question. Given these inputs, it creates risk matrix, assesses if the risk needs mitigation and proposes the treatments for the risks that require mitigation. User if free to allow the system to decide what is the level of risk mitigation needed in order to satisfy the requirements, if decided otherwise, the risk matrix mitigation values can be refined to match the exact required characteristics of a cloud service.

2.2 Creator 4Clouds

Creator 4Clouds offers a graphical modelling environment built on top of a set of targeted UML extensions that allows users to design multi-clouds applications. This set of extensions is collectively called MODACloudML.
MODACloudML provides methodological support for enabling developers to model multi-cloud applications.
MODACloudML relies on three levels of abstractions:

- the Cloud-enabled Computation Independent Models (CCIM) to describe an application as a set of services and interfaces between them;
- the Cloud-Provider Independent Models (CPIM) to describe Cloud concerns related to the application in a Cloud-agnostic way (e.g., its deployment artefacts, QoS constraints, and external services to be reused);
- the Cloud-Provider Specific Models (CPSM) to describe the Cloud concerns needed to deploy the application on a specific Cloud.

Designing a multi-cloud application through the MODAClouds design-time environment is a multi-stage process. First, users specify, through the Functional modelling component, the application architecture and all its functional aspects as well as QoS requirements. In the next stage, on the basis of the high level assets that compose their application, cloud application providers may exploit Venues 4Clouds in order to identify a set of

relevant cloud providers. Designers may then decide, for instance, to select a certain class of database services and certain kinds of computational resources. This process is achieved by QoS engineers and supported by the Space^{Dev} 4Clouds component. The latter can be used to estimate the performance of the identified solution (e.g., response time) and to find a cost optimised multi-cloud deployment configuration. At this stage, an iterative process may start to tune the application design until a suitable solution is identified. The output of this process is a CloudML deployment model that can be used by the application provider to automatically deploy the multi-cloud application.

Exploitable Asset	**Module**	**Component**	**Integrated or External service**
Creator 4Clouds	***MODAClouds IDE***	Functional Modelling Tool	Integrated
		Space^{Dev} 4Clouds	Integrated
		LINE	Integrated
		CloudML 4Clouds	Integrated or external
	Resource Repository	Resource Repository	Integrated
	Feedback Loop	Feedback Loop	Integrated

Table 3:Creator 4Clouds Components

As said before, at design-time, Creator 4Clouds provides first support for interactions with Venues 4Clouds. This interaction consists in exchanging a resource extension container and a Palladio resource environment, describing the high level assets that compose the cloud-application. Creator 4Clouds obtains in return an updated resource container extension that contains the types of providers, services and resources to be used by the application. It can then interact with Space^{dev} 4Clouds to analyse application models. It sends a set of 10 types of Palladio models and extensions to Space^{dev} 4Clouds and obtains in turn an XML file describing the type of providers and resources to be used at the CPSM level to optimize deployment cost and performance.

At runtime, the Creator 4Clouds sends a CloudML deployment model to the Models@Runtime engine, which in turn enacts the deployment. In addition, it sends a high-level description of the application (based on the monitoring rules and QoS constraints and a high level description of the deployment) to the SLA manager, which can in turn provide an SLA id that will be exploited by the Models@Runtime engine to enforce the SLA. Creator 4Clouds can also provide an ER model, which will then transform it to specialized data models and be in charge of their deployment. Finally, during modelling work, the IDE uses the QoS models component to generate monitoring rules from QoS constraints and to validate both of them. Creator 4Clouds provides Tower 4Clouds with the monitoring rules and a description of the operations exposed by the components of the application to be monitored.

Figure 1, summarize the interactions between Creator4 Clouds and the other MODAClouds tools whilst Table 5 describes the different exchange formats.

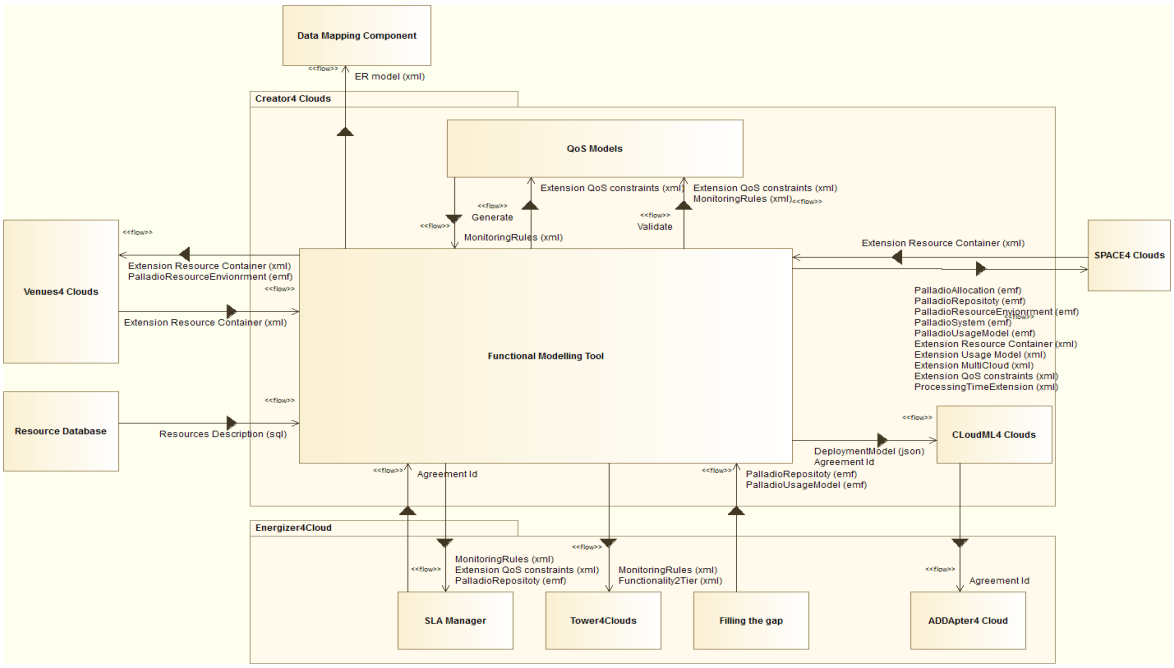


Figure 1 Integration map

Name	File format	Stakeholder	Description
Monitoring rules	XML	POLIMI	Describes what should be monitored on cloud deployed resources and how to react to deviations.
Deployment model	JSON	SINTEF	Describes deployment of the application at CPSM level.
ER model	XML	POLIMI	Describes application data models at CCIM level.
Processing Time Extension	XML	IMPERIAL	Allows users to provide statistical information on execution times.
Functionality2Tier	XML	POLIMI	Lists business operations implemented by each application tier.
Extension Multi Cloud	XML	POLIMI	Defines partitioning of cloud resources in multi-clouds.
Extension QoS Constraint	XML	POLIMI	QoS constraints to be respected at runtime.
Extension Resource Container	XML	POLIMI	Assignment of resources to cloud providers.
Extension Usage Model	XML	POLIMI	Usage model 24h distribution.
Palladio Allocation Model	EMF	Palladio	Allocation of components to resources.
Palladio Repository Model	EMF	Palladio	Description of high level application services.
Palladio Resource Environment Model	EMF	Palladio	Description of deployment infrastructure.
Palladio System Model	EMF	Palladio	Description of high level service instances.

Palladio Usage Model	EMF	Palladio	Description of usage scenarios.
Agreement Id	String	ATOS	SLA Id.

Table 5: Exchange formats

Functional Modelling Tool

Code: http://dev.modaclouds.eu/svn/modaclouds/WP4/InternalDocs/IDE/trunk/
Documentation: http://forge.modelio.org/projects/creator-4clouds/wiki/WikiDeliverable
Deliverable:D4.3.3
License: Apache 2 License

The Functional Modelling Tool is implemented as a module for the Modelio open source modelling tool. It provides a graphical interface, reverse engineering mechanisms and model checking to support the design of MODACloudML models at the CCIM, CPIM and CPSM levels. Besides the code for the functional modelling tool, it contains the necessary code to connect to other design and runtime tools. Communications to other tools are implemented by means of textual files generations and exchange.

CPIM library

Code: https://github.com/deib-polimi/modaclouds-cpim-library
Documentation: https://github.com/deib-polimi/modaclouds-cpim-library/blob/master/README.md
Deliverable: D4.4.1, D4.4.2
License: Apache 2 License

The library introduces a software abstraction layer between applications and cloud services, by exposing Vendor-Independent API. In this way, application code does not need to be rewritten in case of cloud services change. Cloud services supported by the library are relational databases, NoSQL databases, message and task queues, Blob storages, mail services and Memcache service. Moreover, the library enables the Hegira 4Clouds component to seamlessly perform fault-tolerant data migration and synchronization between heterogeneous NoSQL databases.

Space^{Dev} 4Clouds

Code: https://github.com/deib-polimi/modaclouds-space4cloud
Documentation: https://github.com/deib-polimi/modaclouds-space4cloud#modaclouds-space4cloud
Deliverable: D5.4.2
License: Apache 2.0 License

Space^{dev} 4Clouds (System Performance and Cost Evaluation on Cloud) is a tool for specification, assessment and optimisation of QoS characteristics for Cloud applications. It allows users to describe software architectures by means of specific models that include Cloud specific attributes. The tool can be used either to evaluate the cost and performance of a fully described solution (application and Cloud configuration) or to find a suitable (even multi-Cloud) configuration that minimises the application running cost while meeting QoS requirements.

LINE

Code: svn://svn.code.sf.net/p/linesolver/code/trunk/releases/v07
Documentation: http://line-solver.sourceforge.net/#releases
Deliverable: D6.5.3
License: BSD License

The LINE layered queueing network (LQN) analyser interfaces with the Space^{Dev} 4Clouds tool in order to evaluate the performance of the candidate solutions evaluated by Space^{Dev} 4Clouds. Space^{Dev} 4Clouds generates LQN models and passes them to LINE, which provides estimates for Response Time and CPU Utilization related to application components. LINE can also be used as a standalone LQN analyser.

CloudML

Code: https://github.com/SINTEF-9012/cloudml
Documentation: https://github.com/SINTEF-9012/cloudml/wiki

Deliverable: D4.2.2
License: LGPLv3

CloudML is a domain-specific language for specifying deployment models that captures the deployable software artefacts, the middleware required to execute them, and the cloud resources providing the necessary computational resources. CloudML also embed an engine to enact the deployment described in the models or can push them to the Models@Runtime engine.

Resource Repository

Model: http://forge.modelio.org/projects/creator-4clouds/files
Documentation: http://forge.modelio.org/projects/creator-4clouds/wiki/Wiki
Deliverable: D4.3.3
License: Apache 2.0 License

A MODACloudML model representing the most important cloud providers, along with their cloud services and resource characteristics. Its elements are integrated into application CPSM level models created in the Functional Modelling Tool.

Feedback Loop

Code: https://github.com/imperial-modaclouds/modaclouds-fg-analyzer
Documentation: https://github.com/imperial-modaclouds/modaclouds-fg-analyzer/wiki
Deliverable: D5.3.2
License: BSD 3-clause

Feedback Loop is a tool for continuous parameterization of performance models. It provides accurate estimates to parametrise the design-time Quality-of-Service (QoS) models developed in Space^{dev} 4Clouds. Feedback loop aims at obtaining these estimates based on monitoring data collected at runtime, once the application is deployed on the cloud. It implements techniques for the estimation with different monitoring data.

2.3 Energizer4Clouds

MODAClouds Runtime Environment, renamed as MODAClouds Energizer 4Clouds, is a rich execution platform for multi-cloud applications developed with the MODAClouds Creator 4Clouds. Energizer 4Clouds offers solutions to common runtime problems, such as service discovery, packaging, configuration, deployment, monitoring, self-adaptation, and data synchronisation for multi-cloud applications. The platform is divided into two sub-platforms: the monitoring platform, namely Tower 4Clouds, and the execution platform that comprises two exploitable assets, namely ADDapters 4Clouds and Space^{Ops} 4Clouds. The following sections describe these three exploitable assets.

ADDapters 4Clouds

ADDapters 4Clouds is the IaaS and PaaS unified layer responsible for connecting MODAClouds multi-cloud Platform to several underlying cloud service providers at IaaS and PaaS levels. By creating an abstraction layer exposing a Vendor-Independent API to add different underlying providers, ADDapter 4Clouds offers multi-cloud agility, interoperability and bursting among several of the most important IaaS and PaaS providers. It offers a set of portable open-source support services that can be deployed on any IaaS cloud to help package and deploy multi-cloud applications. From this IaaS deployment, these support services can be used to support MODAClouds applications running either on IaaS or PaaS. Support services may be themselves distributed across multi-clouds. Moreover, Support services have been extended in order to allow the deployment of local applications that make use of Glassifish 4.0 Application Server. As part of ADDapters 4Clouds, the CPIM library exposes APIs for access to the following Cloud services offered by the most popular PaaS platforms. The platform is offered to give other providers the opportunity to create a connector and become eligible for being used within the MODAClouds platform.

Exploitable Asset	Module	Component	Integrated or External service
ADDapters 4Clouds	***Data Migration and Synchronization***	Data Migration and Synchronization	External
	Load Balancer Controller	Load Balancer Controller	Integrated
	Object Store	Object Store	Integrated
	Artefact Repository	Artefact Repository	Integrated
	mOS Platform	mOS Image	External
		mOS Package Builder	External

Table 6: ADDapters4Clouds Components

Data Migration

Code: https://github.com/deib-polimi/hegira-components, https://github.com/deib-polimi/hegira-api
Documentation: https://github.com/deib-polimi/hegira-components/wiki, https://github.com/deib-polimi/hegira-api/wiki
Deliverable: D6.5.3
License: Apache 2.0

This component supports data migration within two classes of NoSQL, i.e. graph and columnar. For columnar database, it also supports synchronization between different replicas. It can be started manually by a system operator or automatically by Models@Runtime.

Load Balancer Controller

Code: https://github.com/ieat/modaclouds-loadbalancer-controller
Documentation:https://github.com/ieat/modaclouds-loadbalancer-controller#usage,
Deliverable: D6.5.3
License: Apache 2.0

This component is a RESTFul API for load balancer Haproxy. It allows adding, editing and deleting resources present in Haproxy. Moreover, it also supports defining the load balancing policy as well as the weights. Starting load balancing service can also be triggered remotely.

Object Store

Code: https://github.com/ieat/mosaic-object-store
Documentation: https://wiki.volution.ro/Mosaic/Projects/ObjectStore#Documentation,
Deliverable: D6.5.3
License: Apache 2.0

The Object store provides an alternative to the more traditional locally stored configuration files; it can also be used by deployed services to store small pieces of data.

Artifact Repository

Code: https://github.com/ieat/mosaic-artifact-repository
Documentation: https://wiki.ieat.ro/SilviuPanica/Projects/mOSAIC/ArtifactRepository
Deliverable: D6.5.3
License: Apache 2.0

This component stores MODAClouds artifacts, such as deployment recipes, Maven artifacts, or software packages. It provides a simple REST compliant API for managing the artifacts.

mOS Image

Code: https://github.com/ieat/mosaic-mos-image-builder
Deliverable: D6.5.3
Documentation: https://github.com/ieat/mosaic-mos-image-builder
License: Apache 2.0

This component builds a custom lightweight version of a cloud operating system based on OpenSUSE 13.1. mOS Image represents the core operating system of the mOS Platform.

mOS Package Builder

Code: https://github.com/ieat/mosaic-mos-package-builder
Deliverable: D6.5.3
Documentation: https://wiki.volution.ro/Mosaic/Projects/MosPackageBuilder/Guide
License: Apache 2.0

mOS Package Builder is the component that offers a simple way of building mOS packages. It offers a set of scripts and content description that is needed in order to prepare MODAClouds developed tools to be deployed automatically on OpenSUSE.

Tower 4Clouds

Tower 4Clouds is the integrated monitoring platform of MODAClouds responsible for collecting, analysing, visualizing and storing monitoring information at runtime. It relies on monitoring rules generated from Creator 4Cloudsdefined at design-time and collects metrics from the monitoring components as well as generates statistical inference on the collected data specified in the monitoring rules.

Data Collectors deployment and configuration is described using CloudML and deployed using the Models@Runtime engine together with the application. They are informed about the Manager endpoint by the Models@Runtime engine. Data Collectors can then periodically contact the Manager to synchronize their configuration. The Manager is in charge of offering APIs and a GUI for installing Monitoring Rules and attaching observers to metrics. The Manager instructs Data Collectors on whether, how and where (Data Analyzer endpoint) to send monitoring data. The Data Analyzer is in charge of aggregating, verifying conditions and generating new high level data and is configured by the Manager based on Monitoring Rules. Statistical Data Analyzers can be used to compute more complex aggregations such as statistical estimates and forecast and are enabled automatically through Monitoring Rules.
A model representation of the monitored system is kept alive in the Knowledge Base, which is an in-memory RDF triple store embedded in the Data Analyzer. The history of such model is stored in the RDF History DB automatically. The Manager allows for attaching multiple observers to metrics generated by the Data Analyzer:

- The Metrics Explorer for showing metrics in a dashboard
- The RDF History DB for saving the history of the metric
- Any custom observer

From the last deliverable (D3.5.2), Tower4Cloud has been updated with the following features:

- Embed the knowledge base in the *Data Analyzer* to avoid performance issues by using Fuseki as the DB.
- The deployment model of the applications is updated by the *Data Collectors*.
- A new Web application is developed for installing and visualizing rules, metrics, observers and the model.
- Multiple serialization formats and transfer protocols have been implemented on the data analyzer in order to be able to attach different third party tools without additional adapters (Graphite, InfluxDB, etc.).
- Started to use Grafana (http://grafana.org) as visualization tool, supported by Graphite.

The user manual of Tower 4Clouds is available in [14]. Updated documentation is available at http://deib-polimi.github.io/tower4clouds/docs/.

An updated key components listof Tower4Clouds is reported in Table 8 along with the deliverables containing the detailed description and architecture of these components.

Exploitable Ass	Module	Component	Integrated or External service
Tower4Clouds	***Tower4Clouds***	Manager	Integrated
		Data Analyzer	Integrated
		RDF History DB	Integrated
		Data Collectors	Integrated
		Java App Data Collector (library)	Integrated
		Flexiant Nodes Data Collector	Integrated
		Metrics Explorer	Integrated
		Matlab Statistical Data Analyzers	Integrated
		Java Statistical Data Analyzers	Integrated

Table 9: Tower4Clouds Components

Monitoring SDAs

SDA : SDA-weka
Code : https://github.com/imperial-modaclouds/modaclouds-sda-weka
Documentation: https://github.com/imperial-modaclouds/modaclouds-sda-weka/wiki
Deliverable: D6.3.2
License: GNU GPL

SDA : SDA-matlab
Code : https://github.com/imperial-modaclouds/modaclouds-sda
Documentation: https://github.com/imperial-modaclouds/Modaclouds-SDA/wiki
Deliverable: D6.3.2
License: BSD 3-Clause

Provide high-level statistical aggregations, such as prediction and estimation, from collected metrics that can be reused by the Data Analyzer.

Data Analyzer

Code : https://github.com/deib-polimi/tower4clouds
Documentation: http://deib-polimi.github.io/tower4clouds/docs/
Deliverable: D6.3.2
License: Apache License 2

Processes at high-speed, the monitoring data coming from either the data collectors or SDAs. It interprets, filters and aggregates it based on monitoring rules.

Manager

Code: https://github.com/deib-polimi/tower4clouds
Documentation: http://deib-polimi.github.io/tower4clouds/docs/
Deliverable: D6.3.2
License: Apache License 2

It is responsible for installing monitoring rules, configuring monitoring components, and attaching external observers to requested metrics. It provides both a Web GUI and a REST endpoint to interact with Tower 4Clouds.

Data Collectors

Data Collector : Java App DC
Code: https://github.com/deib-polimi/tower4clouds
Documentation: http://deib-polimi.github.io/tower4clouds/docs/data-collectors/java-app-dc.html
Deliverable: D6.3.2
License: Apache License 2

Data Collector : Flexiant DC
Code: https://github.com/deib-polimi/tower4clouds
Documentation: http://deib-polimi.github.io/tower4clouds/docs/data-collectors/flexiant-dc.html
Deliverable: No
License: Apache License 2

Data Collector : Imperial DC
Code: https://github.com/imperial-modaclouds/modaclouds-data-collectors
Documentation: https://github.com/imperial-modaclouds/modaclouds-data-collectors/wiki
Deliverable: D6.3.2
License: BSD 3-Clause

Update the knowledge base with the monitored resources it is responsible for. Collect monitoring data from monitorable resources and send it to the data analyzers.

RDF History DB

Code: https://github.com/deib-polimi/tower4clouds
Documentation: http://deib-polimi.github.io/tower4clouds/docs/
Deliverable: D6.3.1
License: Apache License 2

Stores monitoring data and knowledge base history in RDF form collected at run time for offline analysis tools developed in WP5.

SpaceOps 4Clouds

Space^{Ops} 4Clouds is a collection of tools designed to enable a cloud application to self-adapt, automatically modifying its deployment, in order to meet predefined objectives and/or satisfy predefined constraints whenever a change in the environment happens.
Software systems have historically been built as according to an open-loop structure. The evaluation of the system-to-be capabilities to meet QoS constraints is performed at design-time. Once the application is deployed, any failure in guaranteeing the required service level would require human intervention, which usually implies costly and time-consuming operations, and a decrease in revenue due to system low performance. To cope with this issue we implemented a set of tools that provides out-of-the-box and independently from the cloud provider a set of adaptive actions based on the feedback control loop, which is very common in control theory and has already plenty of applications in controlled dynamic systems.

The adaptation mechanism in MODAClouds involves the definition (at design-time) and execution (run-time) of policies. Such policies can be activated in several ways; they can be triggered by a monitoring rule registered in

the monitoring platform (reactive policies) as well as by an internal timer (scheduled policies). A Policy defines a protocol to follow to take decisions about adaptation actions. Policies specify the way the application adapts to changes in the environment. The action can therefore be described as a set of modifications of the deployment model or the load balancer.

Currently the Space^{Ops} 4Clouds consists of four major components; Load Balancer Reasoner, Models@Runtime and Cloud Bursting, Self-Adaptation Tester and Auto-scaling Reasoner.

Exploitable Asset	Module	Component	Integrated or External service
Space^{Ops} 4Clouds	Space^{Ops} 4Clouds	Load Balancer Reasoner	Integrated
		Models@Runtime and Cloud Bursting	Integrated
		Self-Adaptation Tester	External
		Auto-scaling Reasoner (MAR)	Integrated

Table 11: Spae4Clouds Components

Load Balancer Reasoner
Code: https://github.com/imperial-modaclouds/modaclouds-load-balancer-reasoner
Documentation: https://github.com/imperial-modaclouds/modaclouds-load-balancer-reasoner/wiki
Deliverable: D6.5.3
License: New BSD

The Load Balancer Reasoner is a component for dispatching requests from end users to application servers following certain load balancing policies. The main policy is weighted round robin and the weights are derived at runtime based on demand estimations on the backend applications.

Models@Runtime and Cloud Bursting
Code: https://github.com/SINTEF-9012/cloudml
Documentation: https://github.com/SINTEF-9012/cloudml/wiki
Deliverable: D6.5.3
License: LGPLv3

The Models@Runtime component is responsible for enacting the provisioning and deployment of multi-cloud applications as well as for adaptation actions such as scaling a certain tier of a cloud application. This component is also able to migrate the application or part of it to a different cloud.

Self-Adaptation Tester
Code: https://github.com/imperial-modaclouds?query=modaclouds-mdload
Documentation: https://github.com/imperial-modaclouds/modaclouds-db-retriever/wiki
Deliverable: D6.5.3
License: New BSD

The Self-Adaption Tester is a workload generation tool known as 'MDload' with the purpose of simulating a set of users to generate requests to a web application in an automated fashion.

Auto-scaling Reasoner
Code: https://github.com/deib-polimi/modaclouds-autoscalingReasoner
Documentation: http://www.modaclouds.eu/wp-content/uploads/2012/09/MODAClouds_D6.5.3_RunTimeEnvironmentFinalRelease.pdf
Deliverable: D6.5.3
License: Apache 2.0 License

This component determines and demands to the Models@Runtime engine the adaptation actions (scaling in and out) that modify the application deployment in response to changes in the runtime conditions. Internally it implements a receding horizon approach and a Mixed Integer Linear Programming (MILP) model.

3 M36 Integrated Platform Testbeds

A number of existing test beds have been used throughout the MODAClouds project, for the development and integration of the components themselves, and for the use of the MODAClouds solution by the case study providers. Expansions and improvements have also been made to support the increasing needs of MODAClouds as integration and experimentation progressed. In the following subsections, we list the most important used testbeds, with a description of their underlying infrastructure.

3.1 Flexiant FCO

The testbed provided by Flexiant has been re-architected to support a multi-cluster environment and recently upgraded to support additional capacity. Figures 2 and 3 detail the Flexiant Testbed.

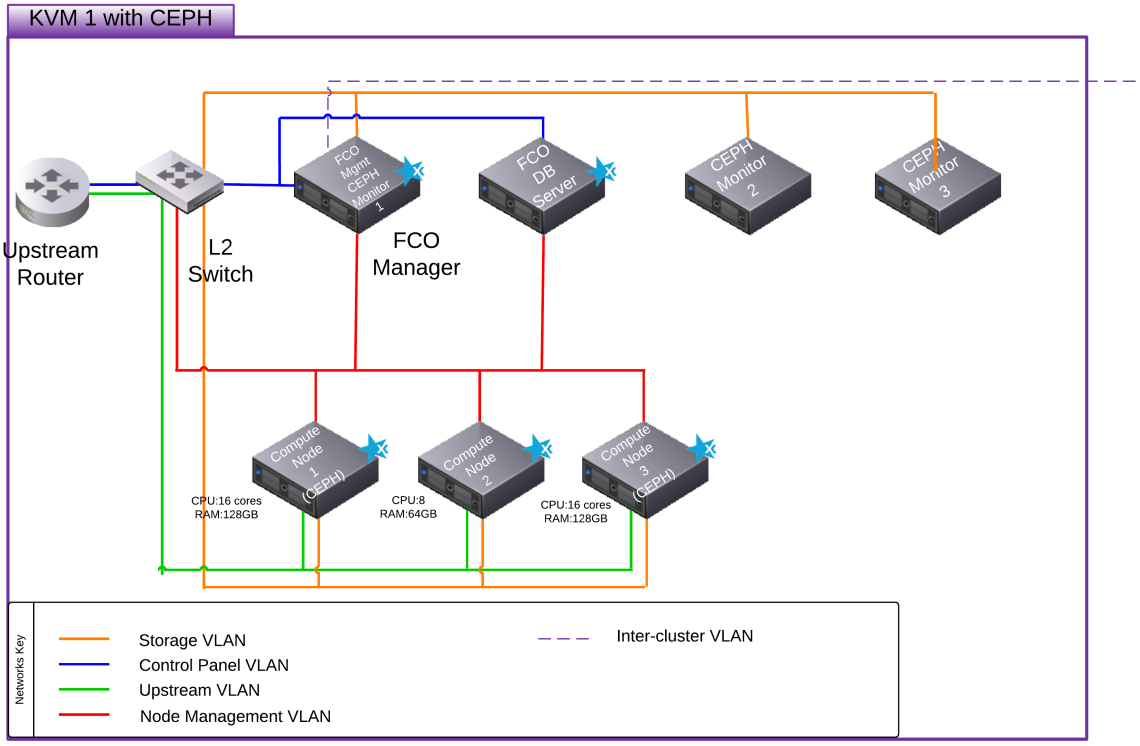


Figure 2 FCO Cluster 1

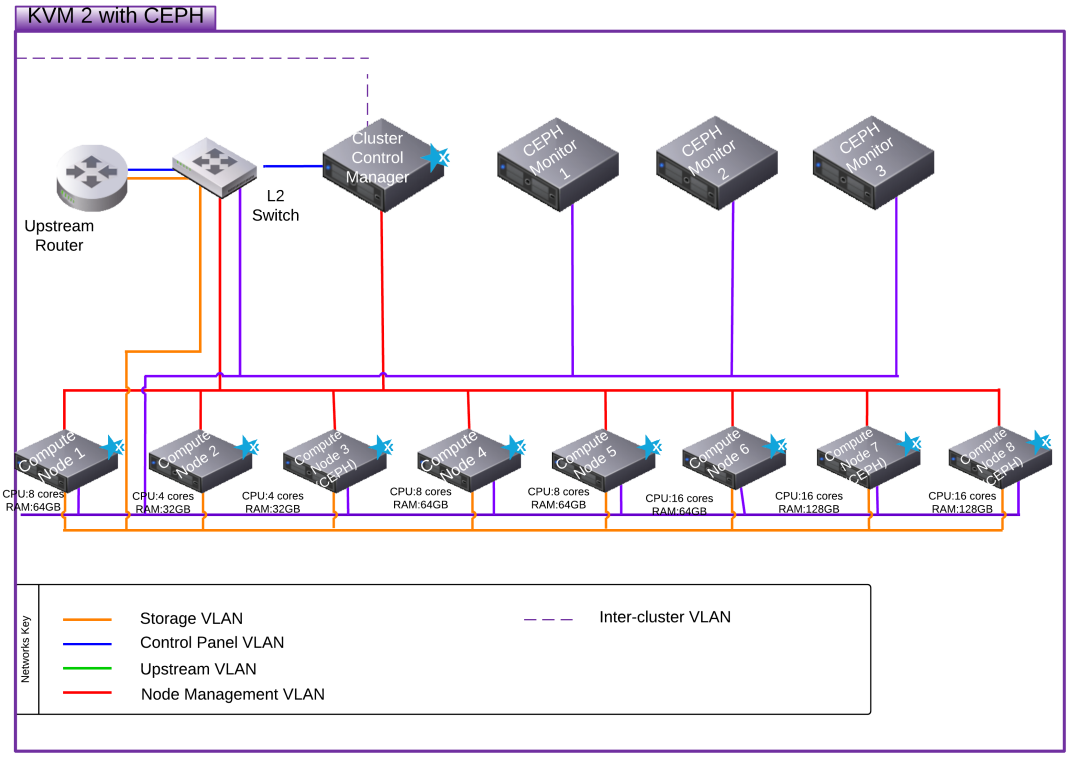


Figure 3 FCO Cluster 2

To provide the MODAClouds project with a closer to industry testbed as possible, a number of improvements have been made to the existing Flexiant testbed. Currently the testbed has been expanded to include 4 new high performance compute nodes. These nodes are each specked with 128 GB of RAM and 16 Cores, which has resulted in an additional 512 GB of capacity, an almost doubling of the previous capacity.
In addition to this capacity upgrade, the backend storage for Cluster 1 has now been updated to use Ceph storage. This has resulted in an increase of speed available to VM's as well as an increase in capacity to the total overall storage for the Cluster.
A further improvement has been the splitting of the FCO management box from the back end Database. The Database has now been placed onto a new separate node. This addition of a new node allows the database to be separate and not share the same resources as the management box. This allows an increase in the total number of API requests as well and improved performance for both the Database and Web console.

With these upgrades completed, the MODAClouds tools have been deployed across the testbed as detailed in the packaging procedures and installation instructions sections.

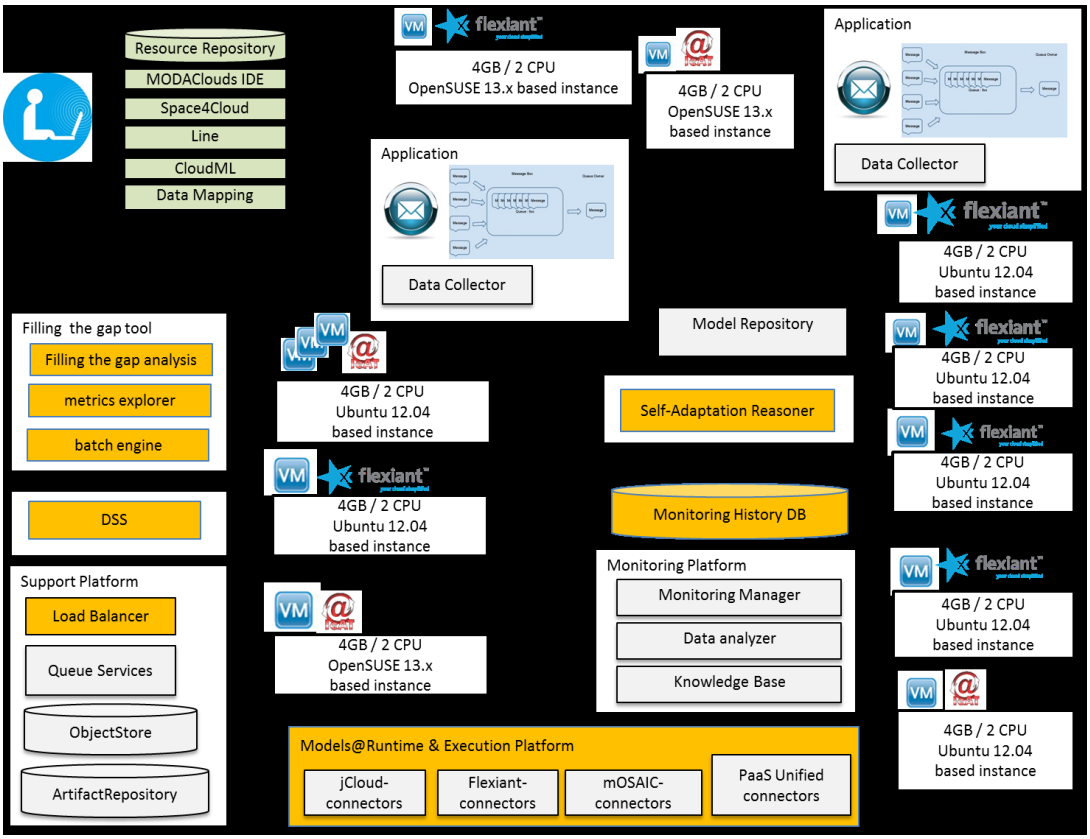


Figure 4 FCO Final integrated testbed

3.2 IeAT Testbed

The testbed offered by IeAT has been upgraded to the Eucalyptus version 4.5 of the cloud stack. The upgrade benefits from the new Amazon EC2/S3 compatibility API update that makes the entire cloud API offered by Eucalyptus fully compatible with Amazons API.

Also, the new version improves the starting process of the virtual machines by lowering the bootstrap process time of cloud images. Moreover, the new user interface has been upgraded by allowing the users to conduct all the management tasks through the UI without the need of using command line tools for managing the cloud resources.

At the hardware level (see Figure 5), the testbed has new compute blades added to support a greater number of virtual machines available for the users. Also, the speed of the storage infrastructure has been increased from 4Gbps fibre channel technology to 40Gbps Infiniband. With this upgrade, virtual machines images are now transferred faster from the images repository to the targeted hosting server.

The upgraded testbed now consists of 8 blade servers with 64 cores and 80GB of RAM memory. Regarding storage capabilities, the cloud environment is linked to a 5TB SAN storage system.
MODAClouds components that are hosted on this testbed have been fully tested and are working as expected.

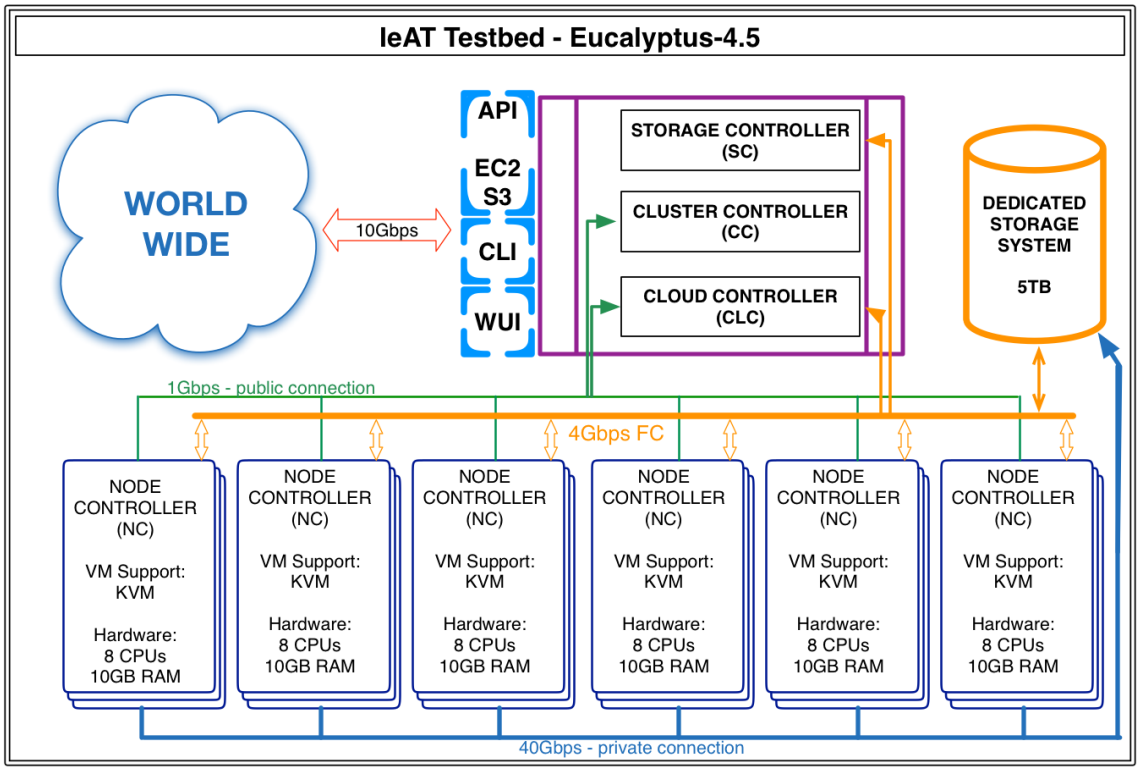


Figure 5: IeAT Final Integrated Testbed

4 MODAClouds runtime platform build instructions

4.1 Building RPMs

While the full detail of how MODAClouds components (services) can be built from RPMs is presented in D3.4.2 and covers management of MODAClouds RPM packages, building the package.JSON descriptor and details on wrapper scripts, it is proficient to summarise the main aspects to accompany and contextualise this final integrated solution.

The developer of each MODAClouds component needs to provide the following artefacts:

- a distribution bundle, namely distribution.tar.gz archive that contains the files needed to run the component
- a descriptor in JSON format, namely package.json, used build the RPM i.e. packing the distribution and the wrapper script
- a wrapper script, namely service-run.bash, that runs the service.

The first step of packing procedure is building the distribution bundle. This is done by the developer and the tools used to do this largely depends on the environment (Java, Python etc.) in which the application has been developed. This artefact together with the wrapper script are next packaged into a RPM package with the help of

the `mos-package-builder.py` script (mOS package builder tool) and the `package.json` configuration file, which is specific to each component. Once the RPM is built, it is deposited under

http://mos.repositories.mosaic-apps.eu/v2/packages/modaclouds/rpm or one of its mirrors. The `package.json` is a file used by the package builder script, which provides all the details needed to generate mOS/OpenSUSE compliant RPM packages, such as general information, external dependencies, license, or other resources.

4.2 Manual Deployment

This section explains how to deploy a MODAClouds Runtime Platform on VMs that have been already been provisioned, producing a running platform with minimal configuration and installation effort. A full set of scripts has been developed to aid the task, allowing not only the installation and configuration of the platform, but also to start services, stop services and check their status. This code is located in the following Github repository: https://github.com/modaclouds/modaclouds-integrated-platform

This method has been successfully used and tested by the case study providers, using VMs and MODAClouds images provided by Flexiant.

The manual deployment can be achieved simply by provisioning VMs from the MODAClouds images then installing, configuring and running the services as described below.

Installation

These installation steps have to be followed for each VM. They are suitable for any OpenSuse13.1 image.
There is such an image available in Flexiant, called OpenSuse13.1_v0.

This image is the same image that can be found in OpenSuse repositories, with some additional configurations:

- admin and root password are the same. Send an email to info@modaclouds.eu to obtain it.
- admin can use sudo without password
- there is a developer account if you want to use it (if not, skip steps related to developer). To switch user to developer: sudo su – developer

You are advised to:

- change root, admin and developer passwords
- upload your public keys
  - ssh-copy-id admin@IP
  - ssh-copy-id developer@IP

First, Git has to be installed from admin user account:

$ sudo zypper install git

From the account you would like to use:

$ git clone https://github.com/modaclouds/modaclouds-integrated-platform
$ sudo modaclouds-integrated-platform/sbin/install-platform.sh <vm-hostname>

The last command adds the mOS repository to the system, updates some system files, installs the components and configures all MODAClouds services (e.g. create database in MySQL for SLA Service).

IMPORTANT:

- It modifies /etc/HOSTNAME and a line in /etc/hosts. Please check the values are right, especially if you had to run the script several times.
- Avoid hostnames with underscore ('_') or dash ('-') characters.

Configuration

After the installation, a little configuration step is needed in each VM to know the IP addresses of the VMs and what services will run in each one. This script must be run every time you want to redistribute the services among VMs, or a node's IP address changes.

$ platform-config.sh <configfile>

You have several configuration files in ~/modaclouds/lib/config-* files. Select the one that better fits the distribution you need and modify it:

- In addresses var, enter a line for each node, with the hostname and its address.
- For each node in addresses var, there must be an instances var prefixed with the hostname.

The file you use will be copied to ~/.modaclouds/config.sh, so the next time you need to reconfigure, you can simply use:

$ platform-config.sh ~/.modaclouds/config.sh

NOTE: The script knows the current node by reading the contents of /etc/HOSTNAME, so it is important to match the names in the config file with the hostname.

Running the components

You have to add the directory that contains the executable files of the modaclouds-integrated-platform to the PATH:

```
echo "PATH=\$PATH:$HOME/modaclouds-integrated-platform/bin" >> $HOME/.bashrc
. $HOME/.bashrc
```

These are the basic commands:

- platform-start.sh: starts all services defined in _platform-env.sh:instance_ids
- platform-stop.sh: stop all services
- platform-status.sh: check status of services
- platform-service.sh: manages a single service (start/stop/status)

The logs are stored in $HOME/var/log. There is one log file per service.

Updating from v1.0 to v2.0

You may have a hostname not suitable for v2 that was not a problem in v1 (remember: no '_' or '-' are allowed).

To modify the hostname, follow these steps:

```
$ sudo su
# HOSTNAME=node1
# echo $HOSTNAME > /etc/HOSTNAME
# hostname -F /etc/HOSTNAME
```

The easiest way to start is using node1 as a hostname. This way, there is no need to change the configuration file.

You also must have an entry in /etc/hosts for that hostname associated to the IP address of the network interface. Do not use localhost. The line to add/modify should be like:

192.168.1.1 node1.localdomain node1

You only have to update your code and run the configuration script using the config file of your needs. In the following, we will use the configuration file using a single VM:

```
cd ~/modaclouds-integrated-platform
git pull
```

platform-config ~/modaclouds-integrated-platform ~/modaclouds-integrated-platform/lib/config1-vm.sh

Check the content of bin/_common.sh file. It should have a VERSION variable greater or equal than 2.0. See the Configuration section for more details about the last step.

4.3 Automatic deployment in mOSAIC

The automatic deployment on mOSAIC is done according to the procedure described in deliverable D6.5.1, chapter 6, more precisely section 6.2.4.

An application descriptor for MODAClouds Energizer 4Clouds deploys all the components of the environment on the mOSAIC platform. The descriptor is uploaded in the integration repository:

-https://github.com/modaclouds/modaclouds-integrated-platform

5 Conclusion

This document is the follow up to the latest integration report document D3.4.2 and the initial version of this deliverable (D3.5.2). It reports the final implementation of the integration plan and a final release, showcasing the MODAClouds solution as it stands at M36. This release and documentation show that the MODAClouds final integration and framework has been delivered on schedule as according to the detailed integration planning.

6 References

[1]. http://en.wikipedia.org/wiki/System_integration
[2]. MODAClouds Integration Plan D3.3.2
[3]. MODAClouds architecture - Final version D3.2.2
[4]. http://www.modaclouds.eu/software/
[5]. MODAClouds Runtime Environment Initial Release D6.5.2
[6]. MODAClouds Integrated Solution – Proof of Concept D3.5.1
[7]. MODAClouds Monitoring Platform - Final Release D6.3.2
[8]. MODAClouds Precision and cost assessment tool - Initial Version D5.4.2
[9]. MODAClouds Runtime Environment - Proof of Concept D6.5.1
[10].MODAClouds Techniques for filling the gap between design time and runtime – Initial version D5.3.1
[11]. MODACloudML IDE – Final Version D4.3.3
[12]. Decision Making Toolkit Requirements and Architecture, and Update on Business Methodology – D2.3.2
[13]. https://github.com/greese/dasein-cloud-flexiant
[14] MODACloudML QoS abstractions and prediction models specification – Final version D5.2.2
[15] http://www.haproxy.org/

A MODAClouds installation and Usage

Details of how to arrive at a running MODAClouds platform can be found here: https://github.com/modaclouds/modaclouds-integrated-platform, and likewise details on running the components can be found here: https://github.com/modaclouds/modaclouds-integrated-platform#running-the-components. There are also installation and usage guides for individual components and as an example, Creator 4Clouds and Space 4Clouds are provided in the appendices below.

A.1 Creator 4Clouds Installation and Usage Guide

Introduction

Due to the advanced nature of the Creator 4Clouds development, we are able to provide granular guidance on the usage of the tool. The Creator 4Clouds is the piece of software that allows users to design cloud applications mostly independently of the cloud provider. The IDE realizes the model driven engineering (MDE) approach leveraged by MODAClouds by means of the MODACloudML modelling language.
MODACloudML allows the users of the IDE to design an application in three levels of abstraction: at the CCIM (Cloud Computing Independent Model) level, the high level set of services that compose the application is defined, along with its data model, architecture and business and QoS requirements. At the CPIM (Cloud-enabled Provider Independent Model) level, a mapping of services onto "cloud resources" and the definition of generic deployment specification and QoS and rules allow the user to simulate and estimate the costs and quality of service of her design very early in the development. Finally, at the CPSM (Cloud-enabled Provider Specific Model) level, those concepts are translated to cloud specific concepts that can be directly translated into running and deployable code.
MODAClouds provides both design time and runtime tool to aid cloud application developers and application providers in creating cloud provider independent applications. The IDE is the integration pivot of the MODAClouds design time tools. It supports users in defining the application model at design time, and in interacting with analysis and runtime tools provided by MODAClouds. At the design group, we include the Decision Support System (DSS) and SPACE 4Clouds. At the runtime tools, we include the Deployment and Provisioning Component (CloudML), the SLA Tool, and the Monitoring Component.

Key Functions of Creator 4Clouds

- The Design Manager component implements the MODACloudML metamodel as a UML profile on Modelio. This is why all other components connect to it in order to access the MODACloudML model. It also implements the necessary user interfaces features needed by a modelling tool: commands for creating elements, MODACloudML specific diagrams and the necessary UI for accessing the other components.
- The Model Transformation component allows the MODACloudML IDE to initialize a CCIM model from models of legacy components; to initialize CPIM model from a CCIM and a CPSM model from a CPIM; and to export the MODACloudML model to the MODAClouds tools.
- The Requirements Management component implements requirements modelling part of the CCIM modelling and the requirements traceability support that is part of the functional modelling tool. It reuses the existing requirements modelling and traceability support in Modelio and extends it with cloud specific functionalities and diagrams.
- The Document Generator component allows the MODACloudML IDE to generate documents from a MODACloudML model. It reuses the Document Publisher module of Modelio.
- The Audit Rules component validates the MODACloudML model being edited on the IDE by means of a rule based engine and a set of validation rules. Rules are classified by severity level: the WARNING rules indicate conditions that may hinder the applicability of some of the MODAClouds tools, whereas ERROR rules indicate conditions that may hinder the applicability of all tools. 47 rules were implemented in Java and delivered along with the IDE.

- The Monitoring rules and QoS requirements management component bridges the gap between the QoS model component provided by WP5 and Modelio low level APIs. It extends the Modelio IDE with commands that allow the QoS and monitoring rules to be validated and with mappings that allow monitoring rules to be generated automatically from QoS constraints.

Creator 4Clouds Prerequisites

MODACloudML IDE requires the following pieces of software:

- JDK 1.6 or higher from http://www.java.com
- Modelio 3.1.2 open-source from here

Along with the following Modelio modules:

- Creator 4Clouds v. 1.x.y from http://forge.modelio.org/projects/creator-4clouds/files
- ResourceModel v. 1.x.y from http://forge.modelio.org/projects/creator-4clouds/files
- Persistent Profile v. 3.0.02 (Available by default on your Modelio installation)
- Java Designer v. 3.0.01 (Available by default on your Modelio installation)

Finally, a MODAClouds runtime instance should be running somewhere, so that monitoring rules, and application can be deployed.

In this tutorial, we will refer to this instance as *localhost.*

Installing Creator 4Clouds

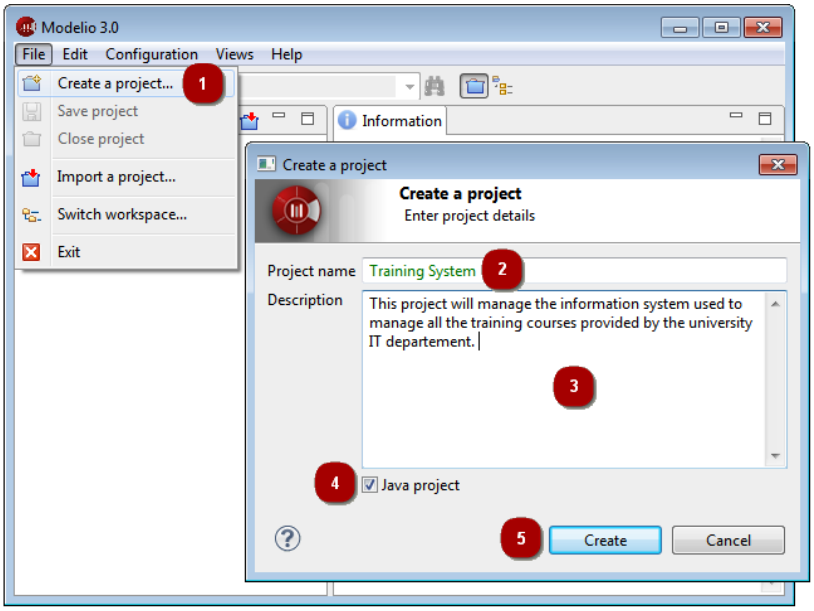
Adding modules from the Modelio modules catalogue

Modelio modules are complementary components, each of which provides specific services tailored to a particular modelling need. Modelio provides a number of modules, all of which exploit a model for a specialized need (for example, documentation or Java code generation). When a module is installed, it provides specific menus, icons and specialized annotations.
To add or remove a module from the modules catalogue, the Add a module to this catalogue and Remove module from the catalogue buttons are used.

1. Open the Configuration / Modules catalogue command.
2. In the Modules catalogue window, click on Add a module to the catalog and use the file browser to select the modules (*.jmdac files).
3. Click on Close button when your catalogue is up-to-date.

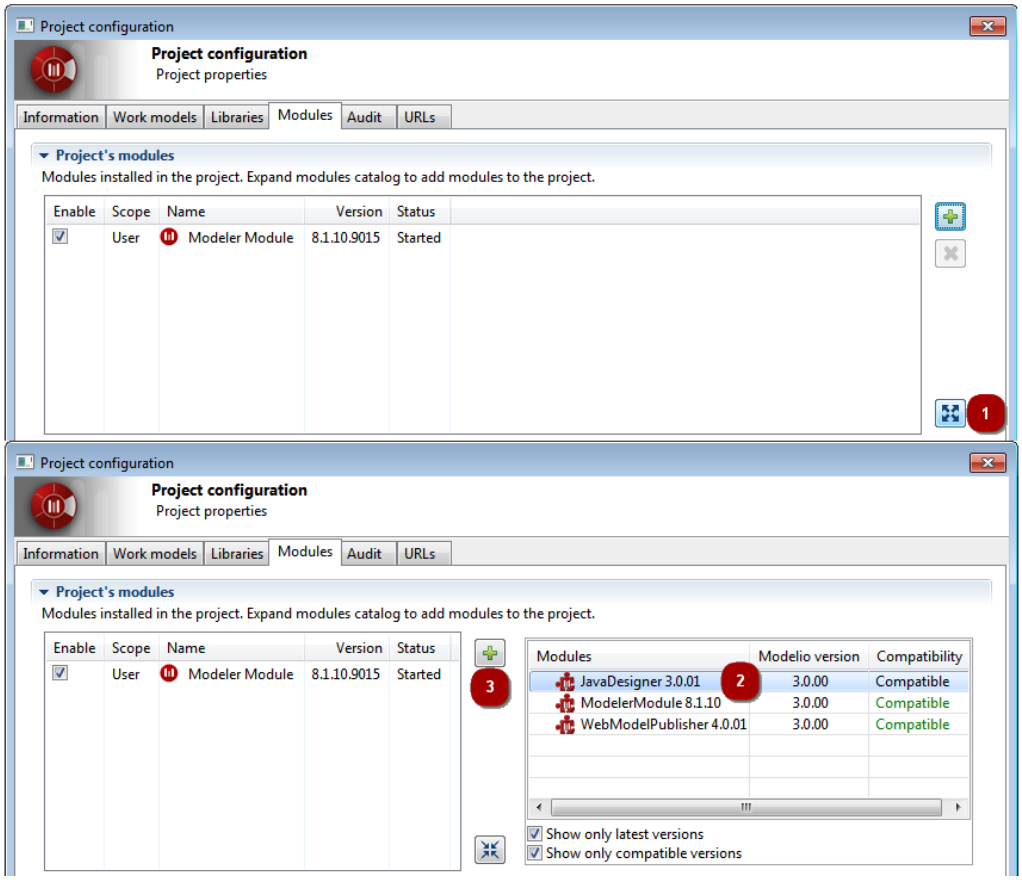
Creating a new project

To create a new Modelio project that is going to contain the MODACloudML model:



1. Click on File\New project.
2. Enter the name of the project.
3. Enter the description of the project.
4. Click on Create to create and open the project.

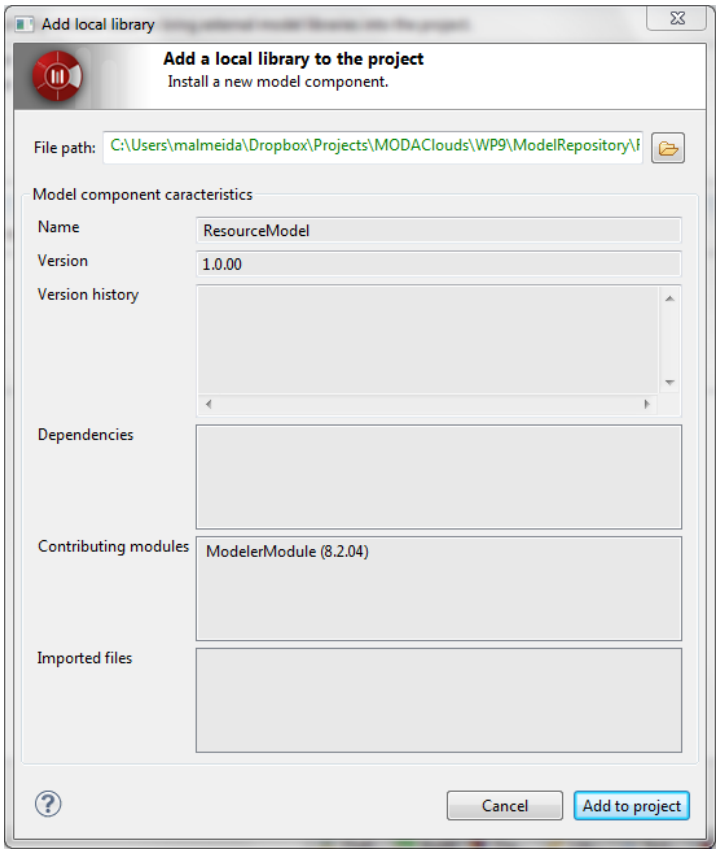
Adding a module to a project


Project configuration
Project configuration
Project properties
Information Work models Libraries Modules Audit URLs
Project's modules
Modules installed in the project. Expand modules catalog to add modules to the project.
Enable Scope Name Version Status
User Modeler Module 8.1.10.9015 Started
1
Project configuration
Project configuration
Project properties
Information Work models Libraries Modules Audit URLs
Project's modules
Modules installed in the project. Expand modules catalog to add modules to the project.
Enable Scope Name Version Status
User Modeler Module 8.1.10.9015 Started
3
Modules Modelio version Compatibility
JavaDesigner 3.0.01 2 3.0.00 Compatible
ModelerModule 8.1.10 3.0.00 Compatible
WebModelPublisher 4.0.01 3.0.00 Compatible
Show only latest versions
Show only compatible versions

1. Open the Module Configuration page in Configuration menu
2. Click on to expand the Modules catalog.
3. In the Modules catalog, select the module you want to install.
4. Click on to install the module in project.

Adding the Resource Model to a Project

1. Open the Configuration \ Libraries menu option
2. Click on the Add button on the Local libraries tab.
3. Select the ResourceModel__1.0.0.ramc file
4. Click on the Add to project button

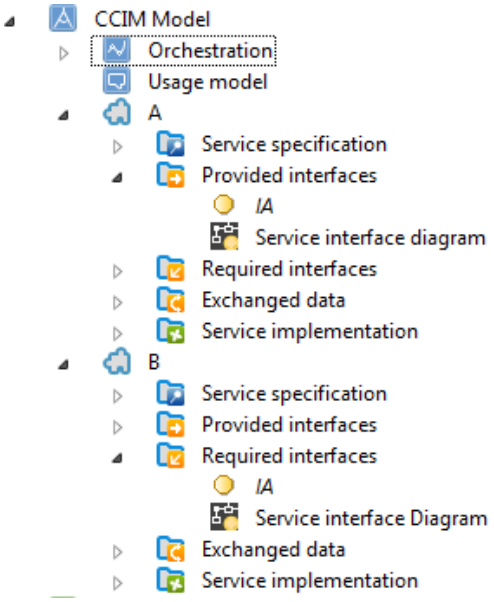


Model an application at CCIM level

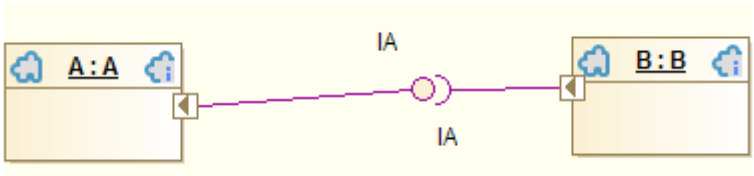
1. Create two services A and B. To do so, right-click on the CCIM model, then select :

   Creator 4Clouds > Elements > Create Service
2. Define interface IA provided by A. To do so, right-click on the Provided interfaces element in A, then select :
   Creator4 Clouds > Elements > Create provided interface
3. Define interface IA required by B. To do so, right-click on the Required interfaces element in B, then select :
   Creator4 Clouds > Elements > Create required interface
4. On the service assembly model, under Orchestration, open the Service assembly diagram.
5. Use the Quick service instance to create instances of A and B, and then connect them.

The final result should look as follows:



Explorer view

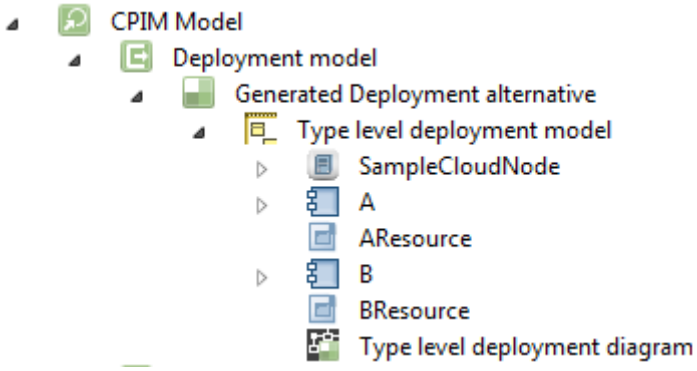


Service assembly diagram

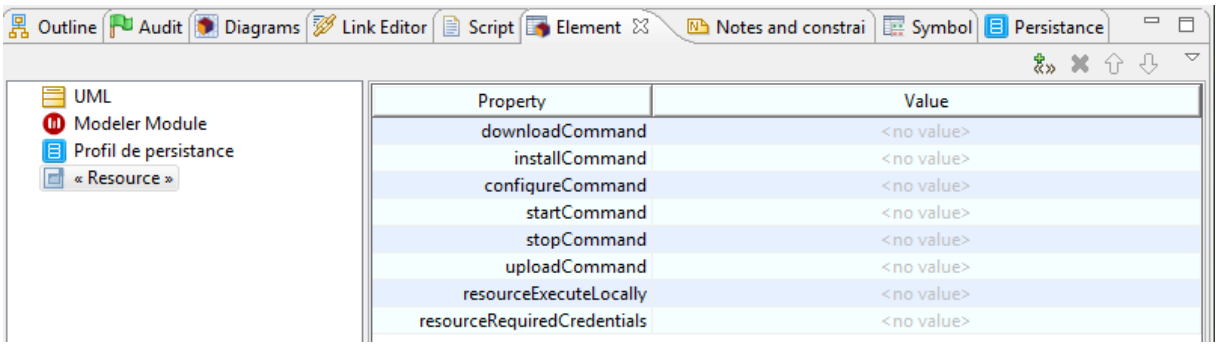
Model an application at CCPM level

1. Generate a CPIM model from the CCIM model. To do so, right-click on the CCIM model, then select :

   Creator4 Clouds > Model transformations > Generate CPIM deployment model
   The generated deployment model should include a type level model only, and should look as follows :

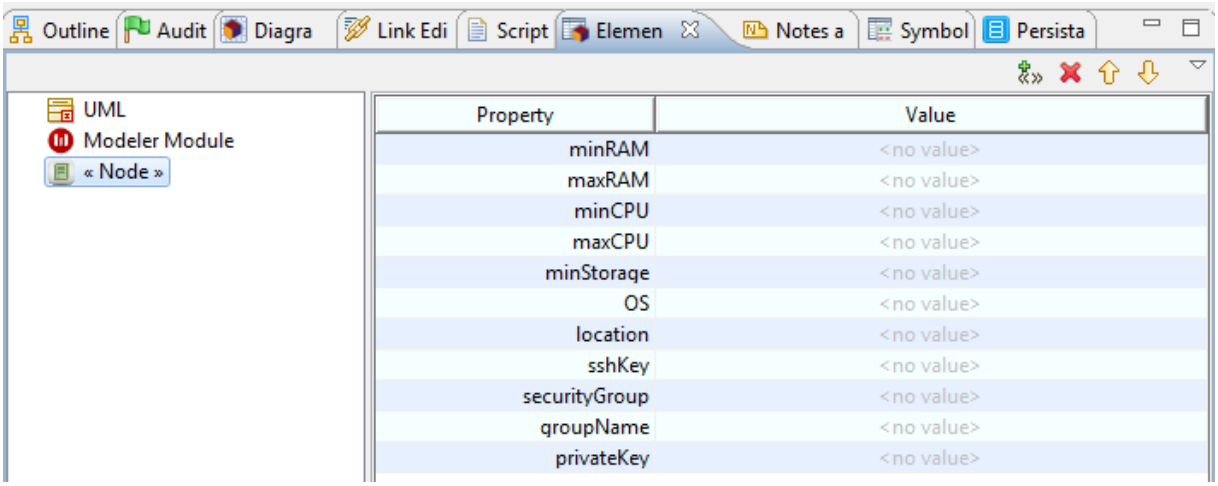
2. Click on each element in the explorer and then use the element view to edit its properties (illustrated below). There you can define the high level aspects of your deployment (ports, connections, OS) and generic deployment scripts to be used during deployment.



Element view at CPIM type level

3. When you are done, generate a instance level CPIM model. To do so, right-click on the deployment alternative, then select:

   Creator4 Clouds > Model transformations > Generate new instance level model from type level model
4. On the instance level model, use the explore view and the diagram view to define the actual deployment architecture. Define in which cloud resources components will be deployed. You can also use the element view to define low level information on your instances (see below).

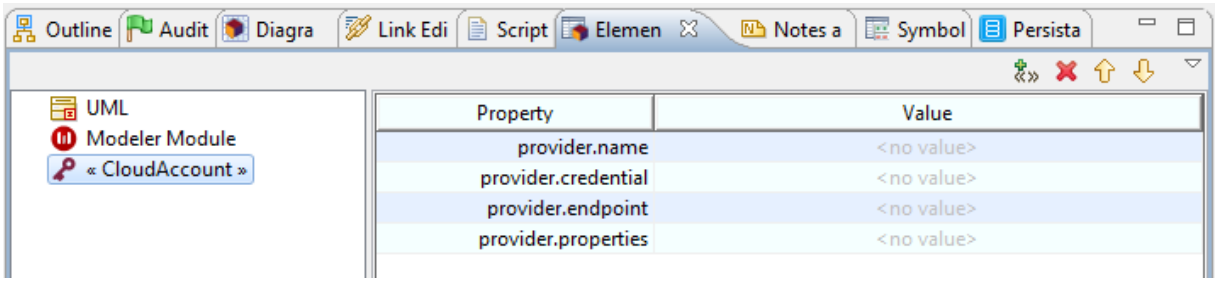


Element view at CPIM instance level

5. When you are done, generate CPSM level model. To do so, right-click on the deployment alternative, then select:

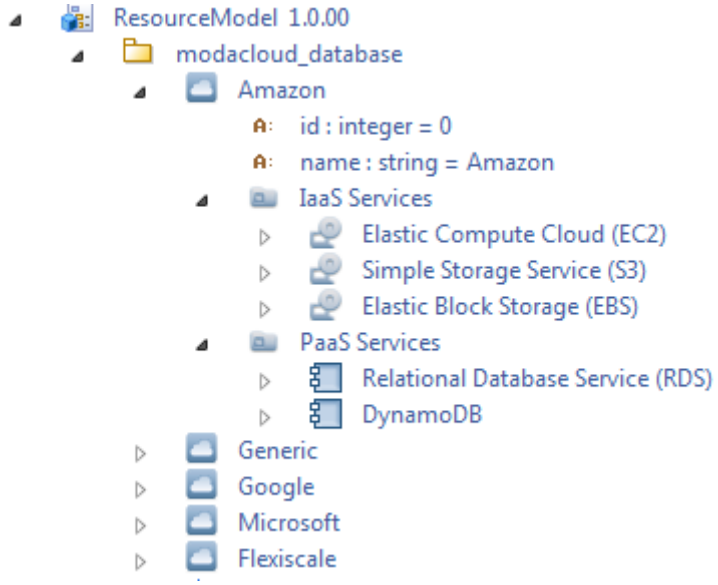
   Creator4 Clouds > Model transformations > Generate CPSM deployment alternative
6. First of all, for each cloud provider you need to use, define a cloud account for it. To do so, right-click on the Cloud account element under the CPSM deployment model, then select:

   Creator4 Clouds > Elements > Create provider account
7. You can also use the element view to define the properties of each provider account (illustrated below). In order to fill these fields, consult CloudML documentation.



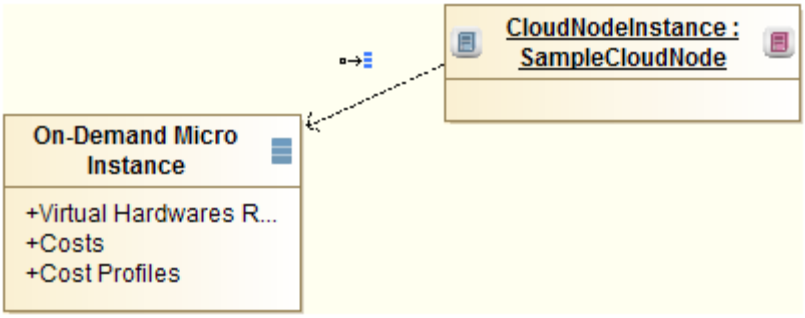
Element View for cloud provider accounts

8. For each provider account, in the elements view, set its Base property, to be the Cloud service from the provider it refers to. Services can be found in the resources model (see below).



Resources model and cloud services

9. On the CPSM models, you should define in which specific kind of cloud resource you want to deploy each node or external component of your application. To do so, open the Instance level deployment diagram, move each element to the diagram and then create a ProviderWorkloadSpecfiication arrow between them. To final result should look like this:



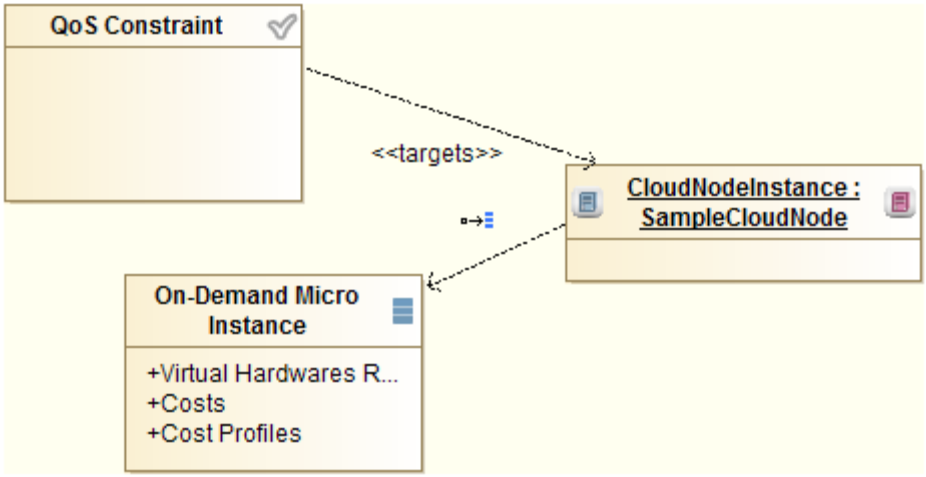
Final instance level deployment diagram.

Model QoS constraints

1. Create a QoS constraint at CPSM level. To do so, right-click on the QoS model under the CPSM model and then select:

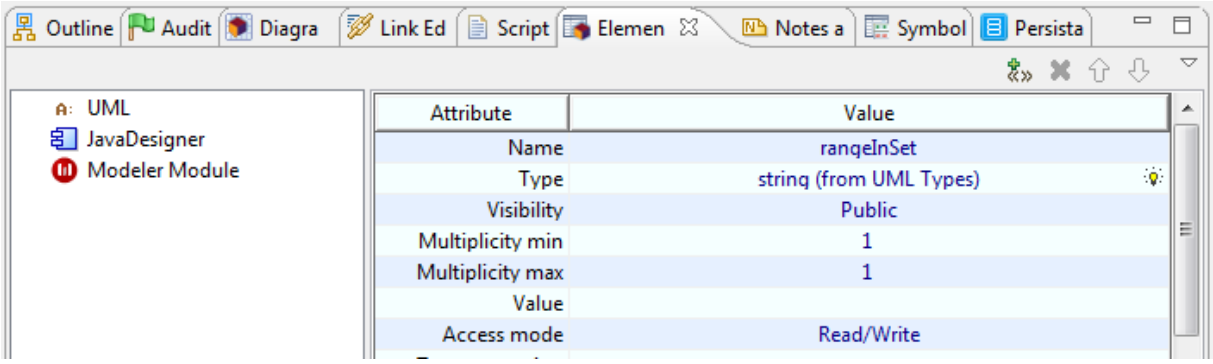
   Creator4 Clouds > Elements > Create QoS requirement

2. Move the created constraint to the instance level deployment diagram. In order to make it refer to a model element, create a dependency link between it and the element. Then add a <<targets>> stereotype to the dependency (see below).



QoS constraint added to the instance deployment model

3. In order to edit the parameters of the constraint, click on each of its attributes and then use the element view to set its Value (see below).



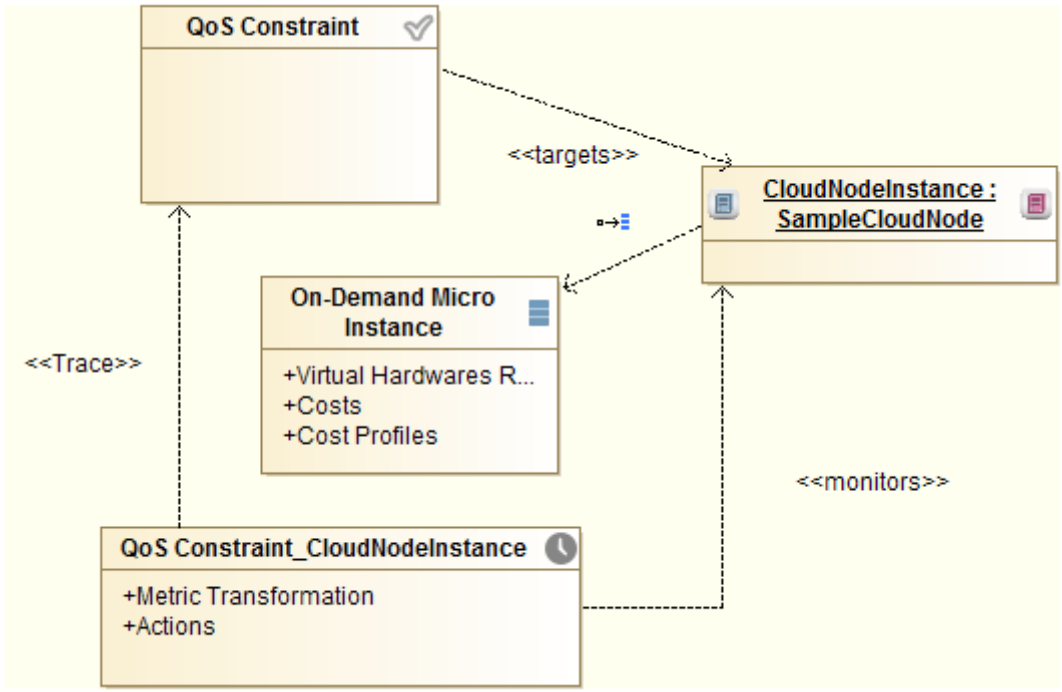
Modelling QoS constraints

Generate monitoring rules

1. Generate monitoring rules from the QoS constraints. To do so, right-click on the deployment model, then select:

   Creator 4Clouds > Model transformations > Generate monitoring rules from QoS constraints

   The generated rules should appear under Monitoring rules, under CPSM models
2. You can also add the generated rules to the instance level deployment diagram. The result should look as follows:

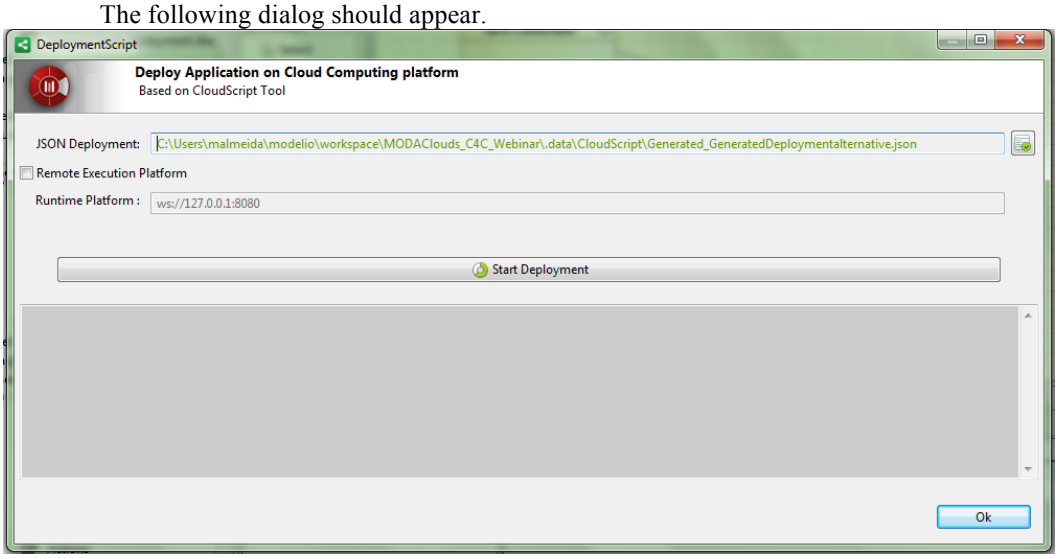


Monitoring rules in CPSM deployment diagram

Deploy application

1. Deploy the application. To do so, right-click on the CPSM deployment model, then select:

   Creator 4Clouds > CloudML 4Clouds > Publish application…
   The following dialog should appear.



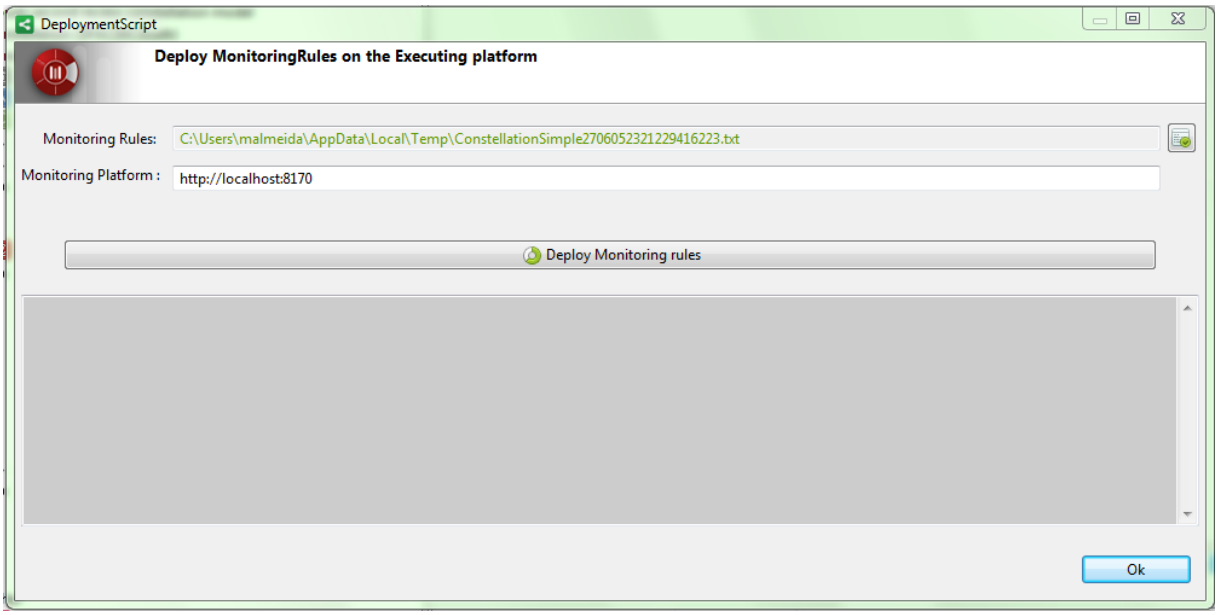
Deploy application dialog

2. Click in Start deployment

Deploy rules

1. Deploy the monitoring rules. To do so, right-click on the deployment model, then select:

   Creator 4Clouds > Tower 4Clouds > Publish monitoring rules…
   The following dialog should appear.



Publish monitoring rules dialog.

2. Set the address of the monitoring platform, then click on the Deploy Monitoring rules button.

A.2 Space 4Clouds Installation and Usage Guide

Features

System PerformAnce and Cost Evaluation on Cloud is a tool for the specification, assessment and optimisation of QoS characteristic of cloud applications. Models of the application, defined in Modelio and exported in the Palladio Component Model formalism with accompanying extensions, are evaluated in order to assess both performance and cost of the modelled solution. The tool is built on top of the Palladio Bench modelling environment but it differs significantly from Palladio since it enriches the modelling capabilities allowing more expressiveness in the definition of the resource environment and the specification of the workload. SPACE4Cloud implements state-of-the-art metaheuristic techniques to effectively and efficiently explore the space of possible alternative configurations. For each configuration involved in the search process, the tool is also capable of evaluating the overall operative cost. However, it makes use of tool LINE for the performance evaluation. More information on the tool can be found in deliverables D5.2.1[1] and D5.4.1[2] of the MODAClouds project.

Requirements

In order to install and run the SPACE4Cloud tool the requirements listed in the table are needed:

Requirement	Version	Where to get it
Palladio-Bench 3.5	3.5	http://www.palladio-simulator.com/tools/download/
Layered Queuing Network Solver (LQNS)	4.5.7.2	http://www.sce.carleton.ca/rads/lqns/lqn-documentation/

[1] http://www.modaclouds.eu/wp-content/uploads/2012/09/MODAClouds_D5.2.1_MODACloudMLQoSAbstractionsAndPredictionModelsSpecificationInitialVersion.pdf

[2] http://www.modaclouds.eu/wp-content/uploads/2012/09/MODAClouds_D5.4.1_PredictionAndCostAssessmentToolProofOfConcept.pdf

LINE	0.5f	https://code.google.com/p/line/

SPACE4Cloud can use either LQNS or LINE as performance engines so at least one of the two has to be installed in the system.

Installation

The installation of SPACE4Cloud over Palladio-Bench 3.5 is performed using the install new software feature of eclipse. It can be accessed as shown in Figure 2 by the menus:

Help -> Install New Software.

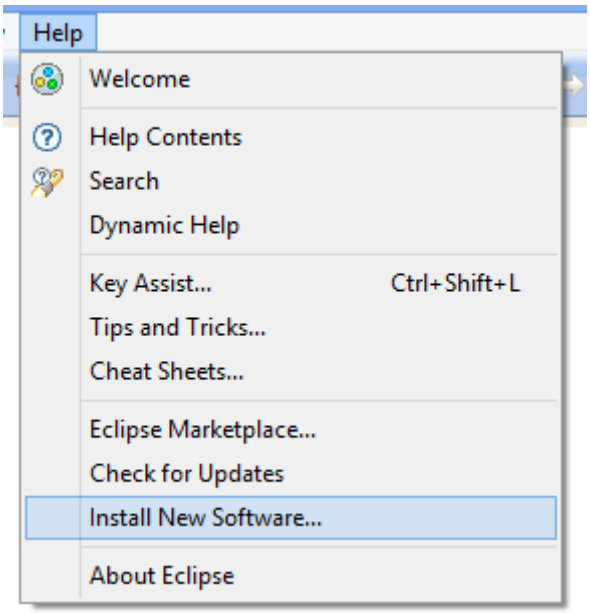


Figure 2

In the "Install" windows click the add a new update site by clicking the *Add* button as shown in Figure 3

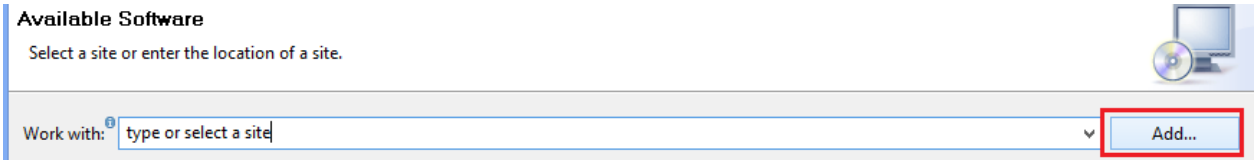


Figure 3

Insert the desired name (e.g. SPACE4Cloud) and use as location as shown in Figure 4:

ftp://home.dei.polimi.it/outgoing/Giovanni.Paolo.Gibilisco/space4cloud/

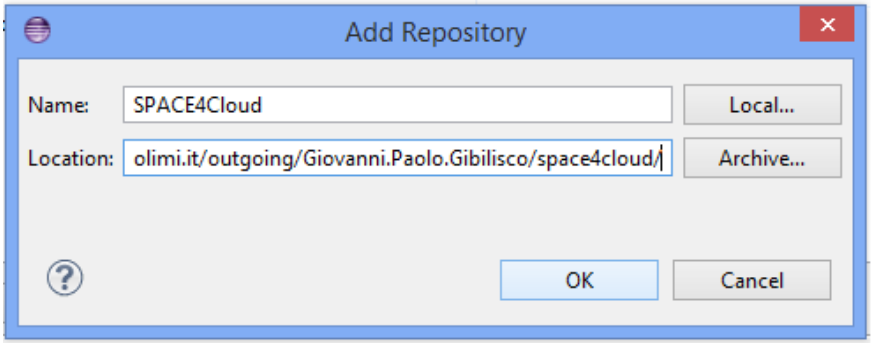


Figure 4

Back in the "*Install*" windows uncheck the "*Group Items by category*" option in order to show the available features, select both the *Space4Cloud* and *Palladio LINE Patch* features and click "*Next*", as shown in Figure 5.

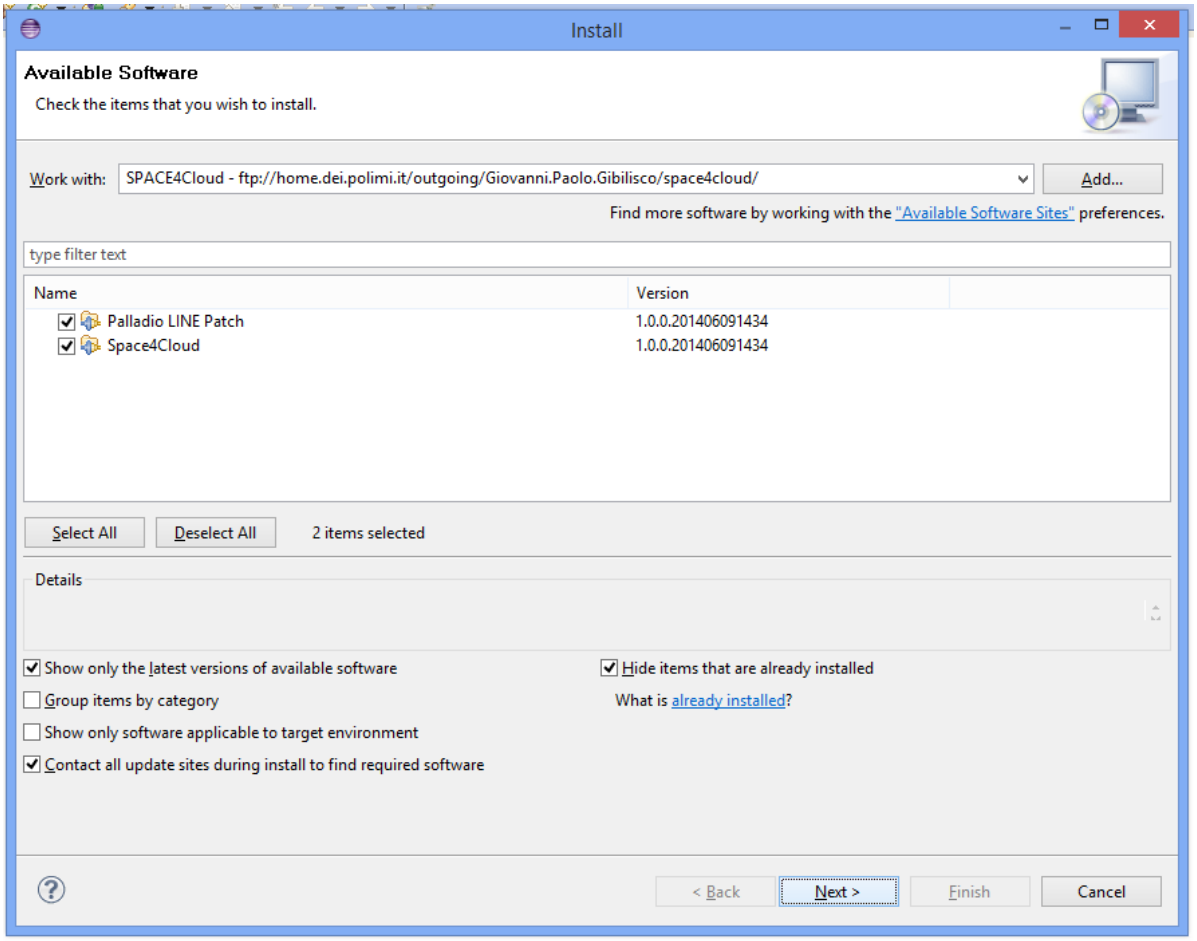


Figure 5

Eclipse will then compute the required actions in order to install patch the Palladio plugin that interacts with solution engines and install the SPACE4Cloud feature. This operation might take a while.

Eclipse will then ask you to confirm the removal of the old version of the Palladio plugin and the installation of the patched one as shown in Figure 6. Proceed by clicking "Next", review the actions in the successive window then click "*Next"* again. Review the license and accept them in the last window.

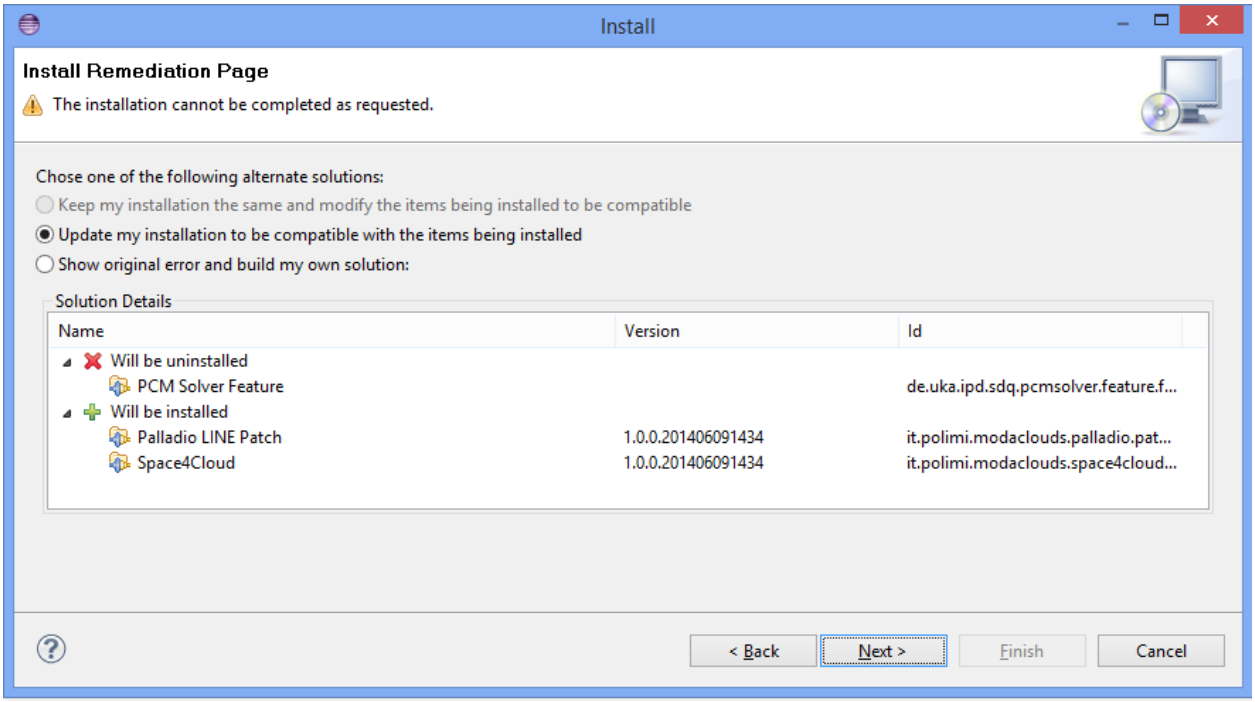


Figure 6

Eclipse will now perform the installation operations and warn the user that it is about to install unsigned content. Click "*OK*" on the warning and proceed with the installation. A restart of eclipse is suggested after the installation has been completed.

Usage

The entry point to the SPACE4Cloud tool in the Palladio-Bench environment is the cloudy shaped icon that appears in the toolbar as shown in Figure 7.

Figure 7

Before starting the tool a project containing the required files has to be created or imported in the workspace. In order to do that the user can make a new project by using the "File->New->Project…" menu in eclipse and then selecting the template for a simple empty project as shown in Figure 8. Then import in the newly created project the needed files.

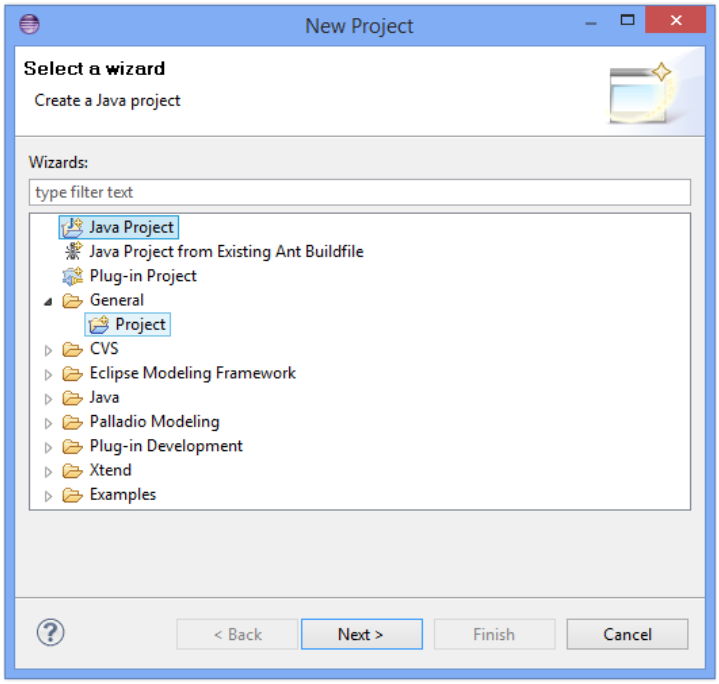


Figure 8

Another option is to download the sample project from ftp://home.dei.polimi.it/outgoing/Giovanni.Paolo.Gibilisco/space4cloudExample and import it using the "File->Import…" menu. Select the "*Existing Projects into Workspace*", as shown in Figure 9, then select as root directory the downloaded one and click on "Finish".

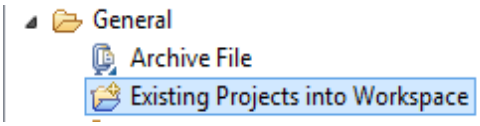


Figure 9

Either by copying the files or by importing the example the project should look like the one in Figure 10.

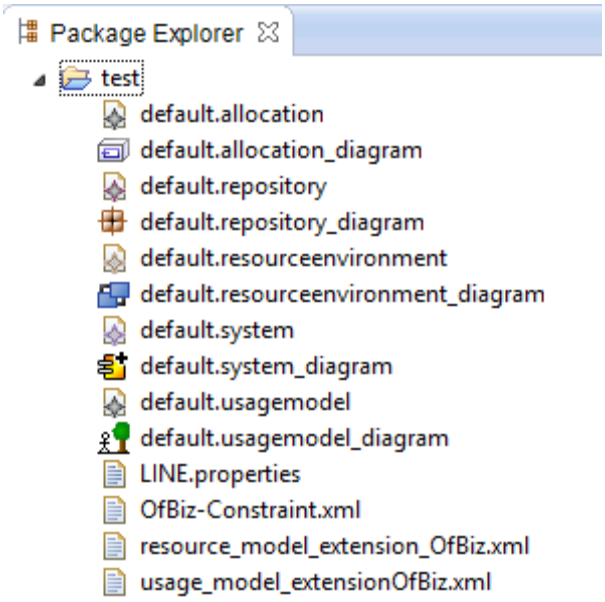


Figure 10

The following table briefly describe the files contained in the example project and their relation to SPACE4Cloud.

File Name	Description
Default.allocation	Allocation model of PCM (**NEEDED**)
Default.allocation_diagram	Visual representation of the allocation model
Default.repository	Repository model of PCM (**NEEDED**)
Default.repository_diagram	Visual representation of the Repository model
Default.resourceenvironment	Resource Environment model of PCM (**NEEDED**)
Default.resourceenvironment_diagram	Visual representation of the resource environment model
Default.system	System model of PCM (**NEEDED**)
Default.system_diagram	Visual representation of the system model
Default.usagemodel	Usage model of PCM (**NEEDED**)
Default.usagemodel_diagram	Visual representation of the usage model
LINE.properties	Configuration file for the LINE performance evaluator (**NEEDED ONLY IF USING LINE**)
OfBiz-Constraint.xml	Extension file containing the constraints modeled by the user (**NEEDED**)
Resource_model_extension_OfBiz.xml	Extension file containing the mapping of PCM resource container to cloud resources (**NEEDED**)
Usage_model_extensionOfBiz.xml	Extension file containing the 24 hour profile of expected incoming workload (**NEEDED**)

In order to run SPACE4Cloud click on the cloudy icon of Figure 7.

A pop up window will ask the user about the desired functionality, currently the supported functionalities are ***Assessment*** and ***Optimization***. The *Robustness* analysis is in an experimental feature that is still unstable.

Both Features will ask the user to load the necessary Palladio models (Figure 11), choose the desired solver (Figure 12), and load the *Usage Model*, *Resource Environment* and *Constraint* extensions (Figure 13).

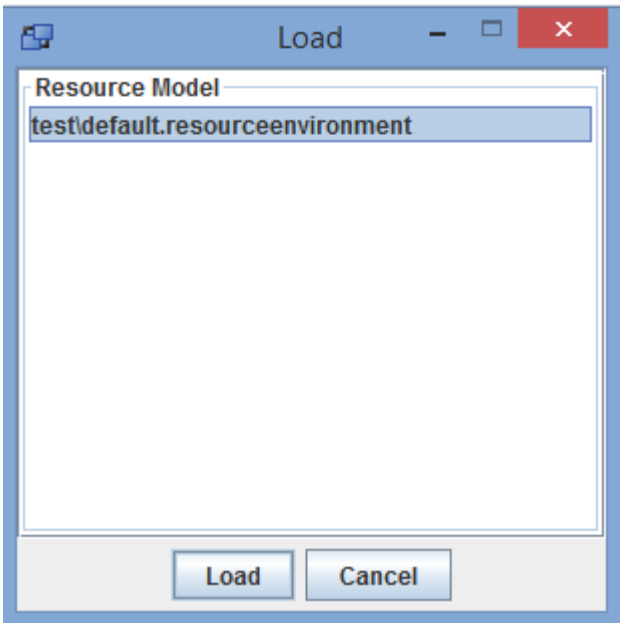


Figure 11

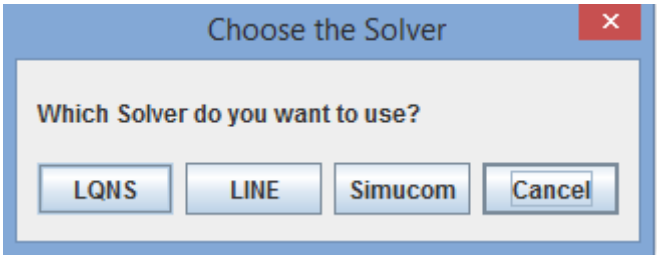


Figure 12

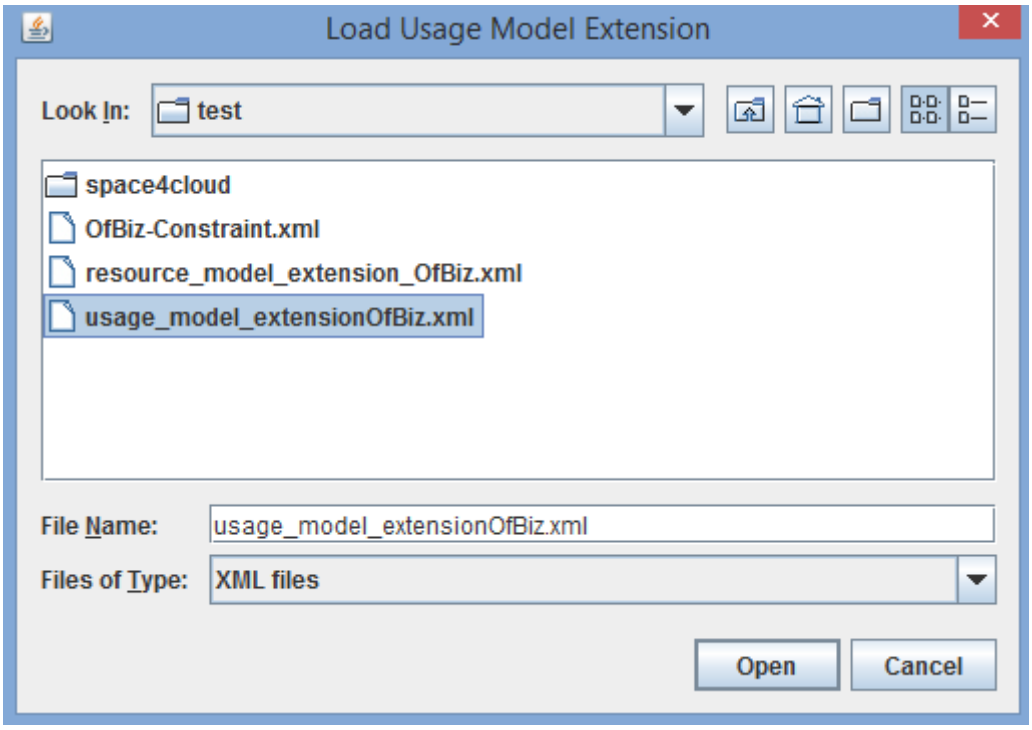


Figure 13

If the user choose to perform the assessment after a short while a windows with the information about the utilization and the response time of resources and functionalities modeled in the application will appear (Figure 15).

If the user selected the Optimization, a pop up will ask the user if he wants to automatically generate the initial solution, since this feature is still experimental we will click on "*No*" and let the tool proceed with the solution described in the extension files. Another Window will appear allowing the user to specify some parameters that are used by SPACE4Cloud to drive the optimization process, as shown in Figure 14. The max iterations and max feasibility iterations heavily affect the time taken in the optimization process as they are used to evaluate the convergence of the algorithm, the memory size is used to store candidate moves and can be increased if the type of available virtual machines is very big. For most of the considered provided the default value of 10 is sufficient. The selection policy define how the algorithm select resources that have to be scaled out when a solution does not meet user defined constraints. Experiments shows that the utilization policy is quite efficient in most situations, but if no constraints on the Utilization of components have been provided other policies can perform better.

Since the optimization can take from few minutes to hours, according to the complexity of the application and the specified parameters, a window will show its progress.

The window show in Figure 16 is divided into three sub figures.

The leftmost shows for each tier the amount of virtual machines used by the candidate solutions. The number on the y axis refer to the sum of all the machines used during the 24 hours. The central image shows the cost of the candidate (in red) and of the best solution found by the algorithm (in blue) per day in dollars. The rightmost window shows the number of constraints that are violated by the candidate solution.

When the optimization process has finished an xml representation of the optimized solution can be found in the *space4cloud* folder of the project in the file *solution.xml*

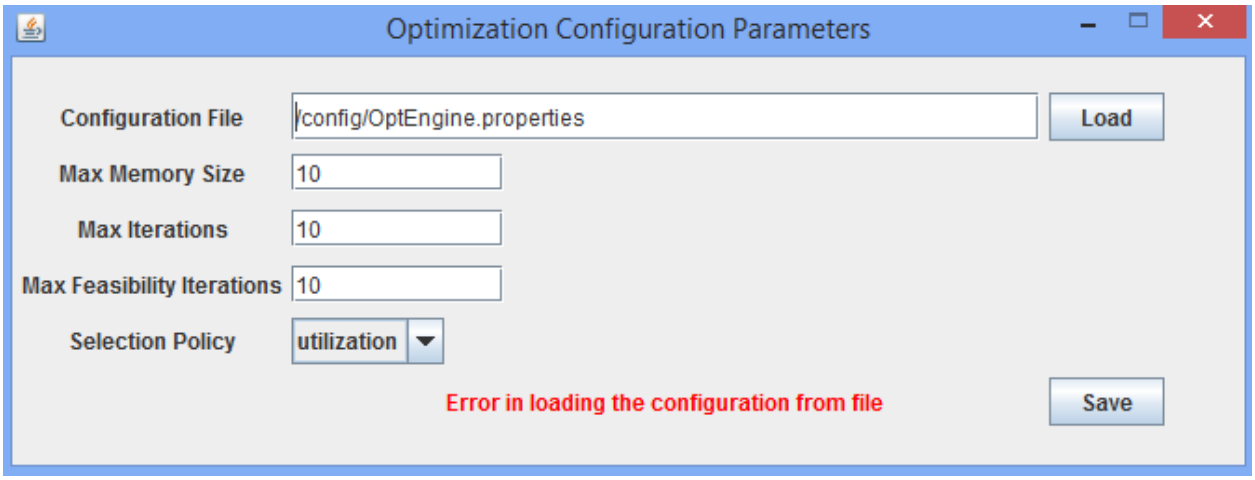


Figure 14

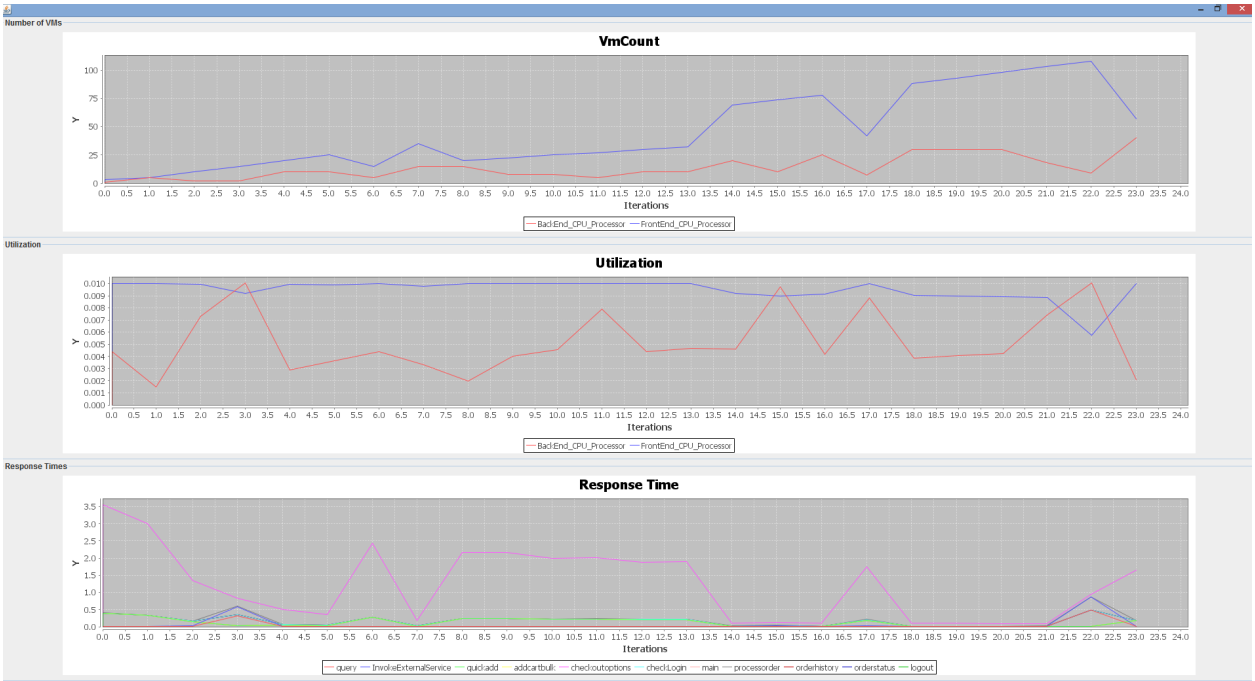


Figure 15

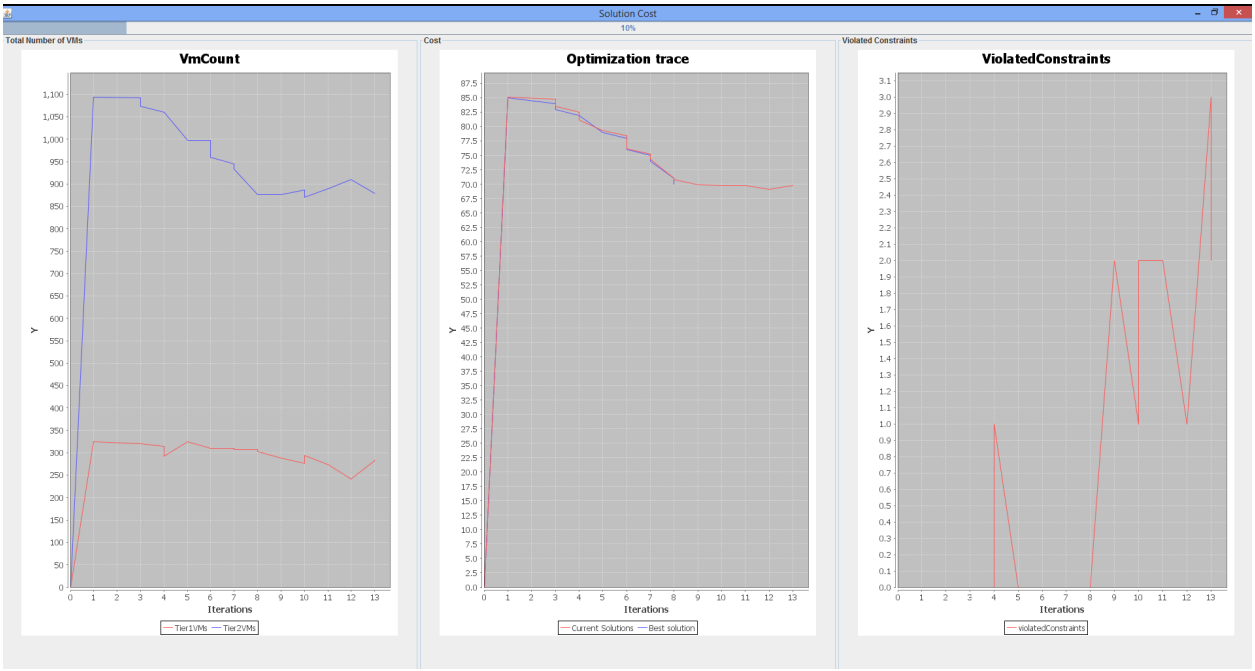

Solution Cost
10%
Total Number of VMs
VmCount
Y
Iterations
Tier1VMs
Tier2VMs
Cost
Optimization trace
Y
Iterations
Current Solutions
Best solution
Violated Constraints
ViolatedConstraints
Y
Iterations
violatedConstraints

Figure 16